NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A COMPUTER SIMULATION APPROACH TO THE STUDY OF EFFECTS OF DECK SURFACE COMPLIANCE ON INITIAL IMPACT IMPULSE FORCES IN HUMAN GAIT

by

David A. Bretz

March 2000

Thesis Advisor: Young W. Kwon
Co-Advisor: Robert B. McGhee

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2000.	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: A Computer Simulation Approach to the Study of Effects of Deck Surface Compliance on Initial Impact Impulse Forces in Human Gait	**5. FUNDING NUMBERS**
6. AUTHOR(S) Bretz, David A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES
The views expressed here are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)*** The Navy's leadership is looking at improving quality of life and reducing long term health problems through the reduction of knee disorders. One proposal for reducing knee disorders is to install more compliant decking The goal of this thesis is to develop a computer model of the human gait that estimates the transarticulation forces in the knee during walking on various surfaces. This model can be used to evaluate the reduction of the heel strike forces during walking when deck surface modifications are made. Previous analytical and computer models of the human gait are reviewed. The major contribution of this thesis is a detailed dynamic model of foot-ground interaction during the initial phase of load bearing in human gait.

14. SUBJECT TERMS Human Gait; Computer Model; Simulation; Deck Surface Compliance; Ground Reaction Forces; Dynamics	**15. NUMBER OF PAGES**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

A COMPUTER SIMULATION APPROACH TO THE STUDY OF EFFECTS OF DECK SURFACE COMPLIANCE ON INITIAL IMPACT IMPULSE FORCES IN HUMAN GAIT

David A. Bretz
Lieutenant, United States Navy
B.S., Marine Engineering, United States Naval Academy, 1993

Submitted in partial fulfillment of the
Requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 2000

Author: David A. Bretz
David A. Bretz

Approved by:
Young W. Kwon, Thesis Advisor

Robert B. McGhee
Robert B. McGhee, Co-Advisor

Terry R. McNelley, Chairman
Department of Mechanical Engineering

ABSTRACT

The Navy's leadership is looking at improving quality of life and reducing long term health problems through the reduction of knee disorders. One proposal for reducing knee disorders is to install more compliant decking. The goal of this thesis is to develop a computer model of the human gait that estimates the transarticulation forces in the knee during walking on various surfaces. This model can be used to evaluate the reduction of the heel strike forces during walking when deck surface modifications are made. Previous analytical and computer models of the human gait are reviewed. The major contribution of this thesis is a detailed dynamic model of foot-ground interaction during the initial phase of load bearing in human gait.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to specifically thank Jesus Christ, the giver of life, for giving me breath and a mind with the ability and desire to use it. I thank my wife, Christy, for her sacrifices and love. I love you! I thank God for bringing us together. Professor Robert B. McGhee for his daily meetings, unceasingly encouraging and positive attitude. I truly enjoyed working with you sir. I look forward to when our paths will cross again. Professor Young W. Kwon for his patience, and belief in me. You never ceased asking the next question. The rest of the great instructors here at NPS. Greg, Susan, and Irma at the Inter-Library Loan office in Dudley Knox Library at NPS. You all are a class act, and quick to get what I needed.

I. INTRODUCTION

A. GOALS

The goal of this thesis is to develop a two-dimensional computer simulation model of a human that has the ability to interact with and simulate mechanical properties of different floor materials. The main intent is to provide a means to determine whether or not a softer or more compliant floor material will alter the characteristics of the initial contact forces during human gait and, therefore, the transarticulation forces within the knee of a human during walking. This thesis discusses the relationship between ground reaction impulsive-forces and knee disorders, namely chrondomalacia and osteoarthritis, and its relevance to the United States Navy. The two-dimensional mathematical model used is a four-mass sagittal plane linkage model with hard constraints in the joints and explicit representation of floor compliance.

This work arises from the desire to improve the quality of life of sailors during and after life on board ships and to reduce the costs associated with lost training, short-term medical care and long-term disability payments. This thesis makes a connection between the hard steel decks of ships, the impulse force felt by the human body during gait, and the above named knee disorders.

B. ORGANIZATION

Chapter II of this thesis outlines the background and motivation, including a look at the knee problems associated with shipboard life and effects of the impulse forces during the initial impact phase of a human gait. Chapter III reviews previous models for computer simulation of the dynamics of human motion, including a postural control model from which this thesis' model was developed. Chapter IV provides an overview of kinematic and dynamic modeling of articulated rigid bodies. The analysis includes "inverse" and "direct" dynamic problems and methods to solve each.

Chapter V describes the four-link human mathematical model used to calculate the results. Chapter VI presents the results of the data obtained from the model. Chapter VII summarizes and draws conclusions from the results, offers suggestions for applying these results to shipboard life, and recommends improvements to the model and further research.

II. BACKGROUND AND MOTIVATION

A. BACKGROUND

1. Knee Disorders in the U.S. Navy

Orthopedic knee disorders account for three of the top 10 medical reasons U. S. Naval personnel separate from active duty service (DeMaio 1999). These knee disorders need to be reduced in order to improve the quality of life, reduce the pain and costs associated with long-term health problems, and reduce the cost of replacing skilled workers. One common injury particularly associated with shipboard life is patellofemoral chondrosis (DeMaio 1999).

Between 1979 and 1981, Helmkamp and Coben (1985) studied the U. S. Navy and Marine Corps enlisted males who entered service in calendar year 1974. Of those entrants, 989 suffered a knee injury that resulted in hospitalization, Medical Boards, and/or Physical Evaluation Boards. Helmkamp and Coben listed the knee problems as loose bodies in the knee, Chondromalacia, other knee derangement, other knee disorders, fractured patella, and dislocated knee. Their study showed that Chondromalacia ranked number one for medical boards and physical evaluation boards for males with a knee diagnosis in this group.

2. Medical Definitions

Hoppenfeld and Zeide (1994) define Chondromalacia as "a pathologic state of softening with subsequent fibrillation (a structural defect seen in articular cartilage that is characterized by splitting of the cartilage surface into fibril-like projections), fissuring (a deep, linear groove, ulcer, or crack with sharp edges), and erosion of articular cartilage." Articular cartilage is a specialized type of smooth and homogeneous cartilage that provides the self-lubricating, low-friction gliding and load-distributing surface of synovial joints. Pathoanatomic knee changes may be classified into the following four groups (Hoppenfeld and Zeide 1994):

Group I: There is swelling and softening of the articular cartilage but the surface remains intact. Also known as Closed Chondromalacia.

Group II: There is fissuring in the softened areas.

Group III: There is fibrillation or fasciculation of the articular cartilage.

Group IV: There is complete erosion of articular cartilage, with exposed subchondral bone.

> Chondromalacia Patella is defined as "pathological changes in the articular cartilage of the undersurface of the patella (knee cap). Clinically, this may be associated with a syndrome of anterior knee pain in which the patient complains of persistent pain behind the patella, especially after sitting with knees flexed for a period of time or when the knee is loaded in the flexed position. The degree of clinical symptoms does not always correlate well with the extent of pathological changes." (Hoppenfeld and Zeide 1994).

3. Biomechanics of Gait-Initial Contact

In studies by orthopedic surgeons, the human stride has been broken down into eight functional patterns or phases (Perry 1992). The first phase is called “initial impact”, also referred to as “heel strike” in the case of a non-pathological gait. Classically, it has been the point at which the analysis of the gait cycle or stride and specifically the stance period begins, and it lasts for approximately 2% of the gait cycle. The gait is also divided into tasks, where the initial impact is found in the weight acceptance task of the cycle. The loading of the limb and subsequently the joints are determined by the posture at the time of initial impact.

Analysis of the first few milliseconds of gait just before the initial impact shows an unbalanced situation where the center of gravity of the body and limbs are forward of hind leg, which has the only contact with the ground. This point becomes a pivot, and the body is temporarily in a state of free-fall. Jacquelin Perry states that this free-fall occurs as the foot that is about to take the impact is approximately 1 cm above the ground (Perry 1992). She also states that 60% of the body weight is loaded on the forward limb in 0.02 seconds.

Various papers quantify this initial impact force in several different ways. Many early force plate measurements do not show this impulse force (Bresler and Frankel 1950, Siegler et al 1982, Chao et al. 1983, and Collins and Whittle 1989b). The absence of the peak force can be attributed to one or more of the following conditions: the sampling rate and/or characteristics of the force plate (Simon et al 1981 and Collins and Whittle 1989b), averaging of the data (Chao et al. 1983), not beginning data collection until the transient has began, or the force not actually being present in the subjects gait or attenuation due to shoes (Cavanagh et al. 1981). The magnitude of the initial impact impulse force varies from none to a peak of 1.25 times body force (Simon et al. 1981). A higher data sampling rate and characteristics of the force plate seem to have the greatest effect on the accurate measurement of the peak force.

All sources agree on the shape of the vertical ground reaction force that follows. It is the typical double hump. The first hump is associated with loading of the heel, a drop follows due to mid-stance knee flexion, and another hump as weight is shifted to the forefoot and push-off sets up the next heel strike. Figure 2.1 shows a typical vertical ground reaction force graph (Collins and Whittle 1989a).

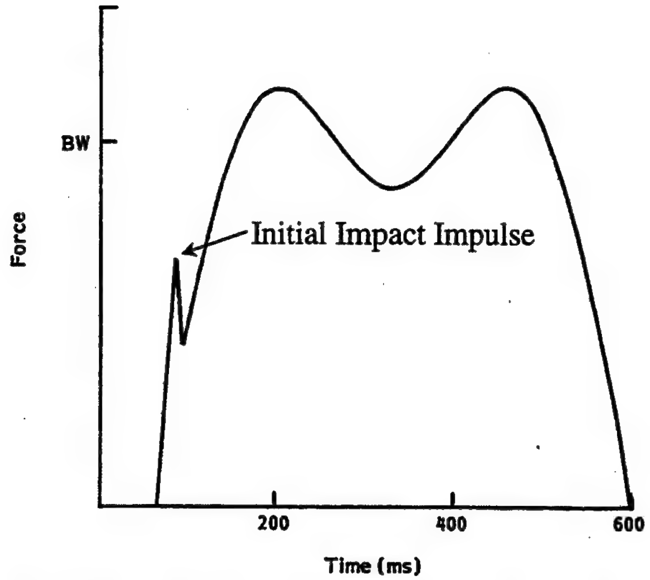

Figure 2.1. Typical plot of ground reaction forces versus time. Modified from Collins and Whittle 1989a.

Simon et al. (1981) conducted an extensive study of the initial impact impulse force. He calls it the "peak dynamic force." He used a Kistler quartz multi-component measuring platform and found the peak magnitude to vary between 0.5 to 1.25 times body weight. The peak was much higher when the subject walked on concrete versus wood.

In order to attenuate the vertical ground reaction forces, the body undergoes several ingenious shock absorbing processes. The first response is "free ankle plantar flexion" immediately following heel impact (Perry 1992). The ankle falls quickly for 10° before the pretibial muscle slows the fall and transfers the motion and momentum to the shank (or tibia or leg). The final two shock absorbing mechanisms are knee flexion and muscle action in the hip during a contralateral pelvic drop (Perry 1992). The body appears to have no natural attenuation for the initial impact impulse force. It is a function of the characteristics of the ground, shoe, heel pad, and gait parameters (velocity, stride length, and angle of foot with respect to ground) (Simon et al. 1981).

4. Effects of Impulse Forces During Walking

Simon et al. (1981) makes the connection between high impulse heel strike transients and osteoarthritis. Collins and Whittle (1989a) review nearly a hundred papers relating to the clinical implications of the impulse force. They show that lower back pain and osteoarthrosis are directly related to the repetitive impulse forces encountered by the human musculo-skeletal system. Although they state that several differences of opinion exist among the experts as to what is actually going on *in vivo* (within the living organism) as to cause the degradation, most experts do agree that the impulse force is a significant contributing factor to whatever may be happening inside the body.

Two methods are given to reduce the initial impact impulse force during walking.

1. Softer surfaces
2. Orthotic inserts.

For the first method, they cite five references that compare soft and hard surfaces. The first reference reported that pigs housed in buildings with hard surfaces were more likely to develop osteoarthrosis than pigs housed in building with dirt floors (Ross 1968). Next, they reference Simon et al. (1981) who show that a harder surface increases the frequencies and magnitudes of the ground reaction forces. The last three references are to E. L. Radin, who investigated the effects of sheep walking on concrete as compared to walking on wood chips. He concluded, "significant changes occur in both cartilage and bone as a result of prolonged walking on hard surfaces" (Radin 1982).

For the second method of reducing impulse forces during walking, Collins and Whittle (1989a) show that orthotic inserts reduced the maximum amplitudes of the bone acceleration by citing Voloshin (1981) and Wosk (1985). Voloshin (1981) shows that artificial shock absorbers in the form of viscoelastic arch support inserts attenuate by 42% the amplitude of the incoming shock waves from heel strike. Wosk (1985) uses these artificial shock absorbers to reduce lower back pain associated with walking.

Guilak et al. (1997) point out that studies have been done that connect repetitive impact loading as related to repetitious daily activity with mechanical changes to the cartilage. These impact loadings have been shown to cause immediate and progressive damage to the articular cartilage. This articular damage, cited by both Collins and Guilak, is the onset of osteoarthrosis and chondromalacia as described by Hoppenfeld and Zeide.

B. MOTIVATION

Obviously, if the impulse forces associated with initial ground impact during human gait are reduced, the medical separations, lowered quality of life, cost of retraining, and long-term cost of disability will also be reduced. Compliant surfaces have been shown to reduce the initial impact impulse force peak. The Department of the Navy has military specifications (Milspecs) thoroughly outlining the requirements as to the quality, durability, and wear, among other things, of acceptable decks and deck coverings for surface ships (S9086-VG-STM-010/CH-634). The question asked is "How does one quantitatively evaluate the reduction of this peak through the adding of more compliant surfaces, such as athletic track material, to the decks of ships in order to reduce the initial impact loading and ground reaction forces in the human gait while at the same time meeting the milspec requirements?"

The quantitative effect of mechanical properties of the surface material on the initial impact impulse force associated with human gait can be determined through experimental measurement or analytical modeling. Three experimental methods include force plate measurements, in shoe force measurements, and limb acceleration measurements. Chapter III of this thesis surveys papers that use experimental or analytical methods in evaluating human gait.

This thesis modifies an existing model of a human gait by adding an explicit foot dynamic model and a spring damper system for the more compliant deck, then lays the groundwork for a quantitative comparison of the magnitude of the initial impact impulse force for different surfaces. This thesis also reviews other models for their potential modification and application to this and other gait related problems.

C. SUMMARY

Chondromalacia, the leading cause of knee problems sent to Medical Boards and Physical Evaluation Boards in the U. S. Navy has been defined. Studies have tied patella chondromalacia and osteoarthritis as well as back pain to the initial impact impulse force during walking. This force's peak is greatest on hard surfaces similar to steel decks on ships. Attenuating the initial impact impulse force by means of a softer or more compliant deck covering needs to be quantified. Various authors have made strides in the area of studying a human gait, which have contributed to the quantification methodology established in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SURVEY OF PREVIOUS WORK

A. INTRODUCTION

Initial impact impulse forces of human gait must be measured or calculated in order to better understand and eventually quantify the effects of material properties, namely, compliance, of a surface on the impulse force. Human gait has been studied extensively, especially in the area of measuring ground reaction forces. Usually these forces and the motion of the body are used to determine joint forces and moments using inverse dynamics models. Additionally, ground reaction forces and center of pressure were measured and used to aid diagnosis of pathological gait. This thesis looks at measuring the ground reaction forces in order to capture the initial impact impulse force.

This chapter reviews a qualitative study of deck surfaces, shoes, and the effects on the body, established methods to measure ground reaction forces, and briefly looks at analytical models used to calculate the ground reaction forces. Additionally, it reviews the postural control model used as the basis for this thesis.

B. QUALITATIVE STUDY OF SOFT SHOES AND SOFT MATS

Hansen et al. (1998), conducted an interesting study in which four combinations of soft shoes, clogs, soft mat and concrete were compared during two hours of standing and two hours of standing/walking work. This type of work appears to closely resemble the watchstanding routine for most personnel on the bridges of U.S. Navy ships. Various measurements were taken during the experiments, including: low back muscle EMG (electromyography), foot volume changes, ground reaction forces, and discomfort felt by the worker. Hansen claims that the effects of working on a soft mat vice concrete are negligible. (This does not agree with the boatswain's mate request for a rubber mat while standing watch at the helm of a warship for several hours.) Hansen's findings point to a greater effect of wearing soft shoes to reducing foot edema formation and heel impact by one half. They do admit that difference in their study between the soft shoe and the soft mat may be due to the "differences in the shock-absorbing characteristics and the thickness of the various materials (Hansen 1998)."

C. DIRECT MEASUREMENTS OF VERTICAL GROUND REACTION FORCES

The initial impact impulse force of heel strike during human gait can be measured directly using force plates and force sensors on the outside or inside of the shoe. By directly measuring the impulse force, characteristics of the impulse can be compared directly for quantitative analysis. An additional technique for quantitatively evaluating the effect of the initial impact impulse forces' effect on the human body, which will not be reviewed here, was completed using accelerometers attached to several points on the body (Bresler an Frankel 1950, Voloshin and Wosk 1981, Morechi et al 1981, Wosh and Voloshin 1985, and Collins and Whittle 1989a). Vertical ground reaction force data collected from miniature insole load cells and data obtained from a force plate in a walkway agree closely, and both show the initial impact impulse force (King and Nakhla 1986).

1. Force Plates

Bresler and Frankel (1950) provide one of the first thorough calculations of the forces and moments in the ankle, knee, and hip during level walking. They used a force plate that measured the location of the center of pressure, torque about the z-axis, and the vertical and horizontal forces applied to the foot. The output was recorded on an oscillograph tape. No indication of the sensitivity of the plate is given, and no impulse is seen on any of the vertical component force graphs during the climb to the first of the typical double peaks.

Today, the use of the force plate is well established, and two of the most commonly used are commercially available through Kistler Instrument Corporation and Advanced Mechanical Technology, Inc. (AMTI). Cavanagh et al. (1981) used a Kistler force platform with a sufficient sampling rate to compare the ground reaction forces of ten male subjects in barefoot, army boots, leather street shoes, and casual suede shoes. Eight subjects showed "well defined peaks between 0.38-0.85 of body weight (BW) occurring during the first 11 ms of contact." The results show that the magnitude of the initial impact impulse force, in order of greatest to least, was barefoot, army boots, leather sole, and crepe sole (of the suede shoes).

Röhrle et al. (1984) use optimization techniques with the data obtained from a force plate-video system to determine forces in the hip, knee and ankle joints. An extensive reference list, detailed description of the gait lab, description of the data processing techniques, and muscle attachment locations would prove to be useful as a reference to someone setting up a gait laboratory or processing data. The data displayed does not show the initial contact impulse force.

Many force plate options are available for various sensitivities, force measuring ranges, natural frequencies, etc. Prices in 1998 varied from $5,000 to $18,000 per force plate. Complete systems that include the force plates and supporting hardware and software range from $15,000 to $24,000. AMTI uses strain gauges and Kistler uses piezoelectricity (Davis and DeLuca 1996). Davis and DeLuca provide an overview with many references of the use of gait analysis in clinical applications.

With more work and decreased cost, a good machine shop can manufacture a force plate using three or four pieziorestive triaxial strain gauges at an approximate cost of $1500 each. Additional equipment required that also should be available in a well equipped laboratory is amplifiers and A-D (analog-digital) converters. The final element would be the software required to process and record the data.

In a study about the reliability of comparing ground reaction force data, Bates et al. (1984) caution that in order to statistically analyze gait data taken on different days, ten or more trials must be taken. Macellari (1996) uses data collected from a compound instrument consisting of a force plate and pressure platform to compare ground reaction forces with a pressure distribution.

Chao et al. (1983) use parametric and non-parametric (Fourier series) representations to statistically analyze a normative database of 148 adults during level walking. They use a Kistler force plate at a sampling frequency of 100 Hz and showed only a small bump where the initial contact impulsive force is seen on other studies. Further, after the Fourier smoothing takes place, no analysis of the initial contact can be made. "It was found that sex-related variation is more significant than age-related variation in adult gait (Chao et al. 1983)." Kerrigan et al. (1998) also studied the difference between gender in joint biomechanics during walking. Although they concluded that there were "more similarities than differences" between males and females with respect to gait kinetics and kinematics, before initial contact "females had greater peak hip flexion and less knee extension (Kerrigan et al. 1998)." Initial impact impulse force does depend on the angle of limb approach (Simon et al. 1981). This thesis will assume no difference between sexes or ages.

2. In-Shoe Force Sensors

In-shoe force sensors with inexpensive and lightweight recorders have been used for collecting two to three hours of ground reaction force data. This data can then be downloaded into a computer and analyzed. Such in-shoe devices can be used to measure the initial impact impulse force as a subject walks on various materials in order to quantify the change in the impact force. In-shoe measuring devices are particularly appealing due to their low cost, portability, and the fact that a sensor inside the shoe is closer to the heel, and may be more representative of what is happening there, because the heel of the shoe effects the impulse force. This procedure would be especially useful to analyze several sailors aboard a modern warship to determine where the greatest forces occur and, therefore, where the most critical parts of the ship for compliant surface coating are found.

Abu-Faraj et al. (1996) used Force Sensing Resistors (FSR) and an on-subject data recording device weighing 350 g to record two hours at 40 Hz of in-shoe pressures for the purpose of evaluating metatarsal and scaphoid pads. Dingwell et al. (1997) use an insole Personal Force Monitoring Device (PFMD) for recording ten hours of continuous ground reaction forces at a sample rate of 25 Hz. All on-person parts weigh 1.5 kg and "consist of a pair of capacitative force monitoring shoe insoles and signal amplifier (Electronic Quantification, Inc.), a Tattletale microprocessor (Onset Computer, Inc.) with a four megabyte PCMIA card for data storage, and an elastic waist belt to which these electronics are attached" (Dingwell 1997). The authors outline a method for quantifying the load bearing activities of the subjects.

D. ANALYTICAL MODELS OF GAIT

Another option for the quantitative evaluation of initial impact impulse forces, and the direction chosen in this thesis, uses an analytical model. Several analytical models exist from finite element analysis to inverse kinematics and direct dynamics for the analysis of a human gait. Chapter V of this thesis addresses the specific calculations of one type of analytical model, namely the recursive Newton-Euler direct dynamics model.

Many computer simulation models exist for the analysis of the human gait (Abdelnour et al. 1975, Siegler et al. 1982, Pandy 1987, Pandy and Berme 1988, Amirouche et al. 1990, and Zanchi and Cecić 1996, among others), but few, if any, look at the first few milliseconds of heel impact. Abdelnour et al. (1975) use experimental gait kinematics data and Lagrange's form of d'Alembert's principle (Kane's equations) to compute joint forces and moments and ground reaction forces. A small peak does appear on the z-axis normalized foot force (vertical ground reaction force) figure; however it is not discussed in the paper.

Amirouche et al. (1990) present "an all purpose algorithm used in the analysis and simulation of human locomotion." An initial configuration is set up, and different constraint equations are used to analyze various human movements. Ground reaction forces are computed without force plates. Kane's equations are also used. The motion of the human is only tested for the swing phase of gait, and no analysis is made of the initial impact.

Koozekanani et al. (1983) extended a recursive Newton-Euler inverse dynamic model to direct dynamics. They demonstrate the technique on a simple open serial chain model for planar postural dynamics. The shank, thigh, trunk, and head angles of the simulation agree very closely to the experimental results. Pandy (1987) and Pandy and Berme (1988) apply a recursive Newton-Euler inverse dynamic model to the two phases of human gait (single and double support) separately, and no mention is given to the initial impact impulse force that connects the two phases.

The models of human gait are typically subdivided into double limb support or stance and single limb support (Pandy and Berme 1988). During double limb support, both feet are on the ground and equations are used to simulate a closed kinematic chain, while an open kinematic chain is simulated during single support.

T. McMahon et al. (1979), built an analytical expression to "predict separately the effect of track compliance on step length and ground contact time." These effects were functions of track stiffness. The results were compared to experimental results. The results also showed that initial contact impulse force reaches nearly five times body weight during running on a hard surface, but that more compliant tracks can show a "marked attenuation (McMahon 1979)." Although this study was performed on runners, the impulse does exist during walking at various speeds.

E. OTHER REVIEWS

Wilson et al. (1997) used principal component analysis (PCA) on existing recorded data to determine statistical correlations between walking, running, rigid and mat footfall surfaces, gender, and arch indexes. PCA showed no effects of the different footfall surfaces. However, Wilson cautions that sometimes PCA may mask changes in response parameters, and a 24% reduction of the vertical load rate was seen for barefoot subjects walking on the matted surface as compared to the unmatted surface.

F. SUMMARY

A qualitative evaluation of the effects of surface compliance on the initial impact impulse force of a human gait requires an accurate measurement or determination of characteristics of the impulse force. These characteristics can be made through direct measurements using force plates or miniature load measuring devices either in or on the outside of the shoe. Analytical models may also be used to estimate the impulse force and use this data for a qualitative comparison. An analytical model was chosen for this thesis due to cost and time constraints. Chapter IV of this thesis reviews mathematical models that most directly apply to the model used in this thesis. Although an analytical model cannot simulate the gait exactly, especially the initial impact, an analytical approximation would be useful for checking relative changes in the initial impact impulse force due to deck compliance.

IV. MATHEMATICAL MODELS

A. INTRODUCTION

In order to begin to model human gait to quantify initial impact forces, an understanding of dynamics and methods used for these calculations are essential. Dynamics is the study of the motion of bodies. It is divided into two areas of study, kinematics and kinetics. Kinematics looks at the motion of the body (position, velocity, and acceleration) without considering the forces on the body that caused the motion (Craig 1989). Kinetics relates the forces and moments acting on a body to its resulting motion.

A kinematic model calculates a rigid body's position (Cartesian space) and orientation (roll, elevation, and azimuth) given an adequate description of the body in terms of body segment lengths and joint angles. This type of calculation is called forward kinematics. Inverse kinematics does the reverse, finding joint parameters from the given orientation and position of body segments. Kinematics also includes the relationship between position, velocity, and acceleration within and through the given coordinate system(s).

A "forward" or "direct dynamics" problem calculates the body's motion from the given forces and moments acting on the body. "Inverse dynamics" calculates the forces or moments required to cause a given motion of the body. Dynamics in general is the union of kinetics and kinematics to understand and describe the relationship between position, velocity, acceleration, and orientation (in any coordinate system) based on the body's parameters and the forces and moments acting on that body.

A human dynamic model may be as simple as an inverted pendulum with one degree of freedom or a more realistic model with 200 degrees of freedom (McGhee et al. 1979). The complexity of the kinetic and kinematics calculations increases tremendously as the designer attempts to three dimensionally model individual muscles and the compliance and damping associated with joints, vis-a-vis a two-dimensional model of planar forces and moments with hard constraints in each of the joints.

B. KINEMATIC MODELS

Kinematic models of human gait enable the comparison of a healthy gait to a pathological gait. Inverse kinematic models are used by recording the position and orientation of subject through film or video recording to calculate angles, timing, and phase relationships. "Through advanced reasoning, which correlates the patients performance with normal phasic function, the primary defects can be differentiated from substitutive actions" (Perry 1992). Someone with enough experience or insight can be aided in a diagnosis of a pathological gait from analysis of kinematic data. Also, gait improvements during rehabilitation can be quantitatively made from kinematic data.

Ju and Mansour (1988) used foot kinematic data to construct a planar foot model for use in a simulation involving the double limb support phase. Kinematics is also fundamental to dynamic models.

C. DYNAMIC MODELS

1. Direct Dynamic Models

Direct dynamics solve for motion (orientation, position, velocity, and acceleration) of a rigid body based on the force and moment inputs acting on that body. Accurate forces and moments must be obtained in order to be used for calculations in a direct dynamics human simulation. These forces and moments can be obtained from an inverse dynamics model, feedback control, or optimization techniques. Most models use the forces and moments obtained from inverse dynamics models as described in the next section (Abdelnour et al 1975 and Ju and Mansour 1988 among others).

2. Inverse Dynamic Models

Inverse dynamic models use the position, velocity, acceleration, and ground reaction force data obtained from video recording equipment, force plates, and appropriate computers and software, usually within a gait analysis laboratory, to determine the forces and moments at the ankle, knee, and hip (Bresler and Frankel 1950). These forces and moments can then be applied to a direct dynamics computer simulation or mathematical model of an articulated human body to determine the position, velocity, and accelerations. This movement is then compared to the original human movement to determine the accuracy of the model. This type of analysis has gone beyond purely academic exercises of understanding human gait to clinical analysis of pathological gait and posture. The nature of a problem evidenced by weak or over-used muscles is better understood by studying the pathological gait, and comparing it to a healthy gait.

D. ARTICULATED RIGID BODY DYNAMICS

A “link” is a rigid body connected to other link(s) in a chain by joints to form a "manipulator" or serial linkage mechanism (Craig 1989). Craig (1989) describes a link with four parameters: link length (a), link twist (α), link offset (d), and joint angle (θ). A human body can be simulated as an n-link open chain articulated planar mechanism. The chain is “open” because the end-effector (head or torso) has no forces or moments acting at its free end. A closed chain has both ends constrained such as when modeling both legs during the double stance phase of gait. The mechanism is planar because the joints are modeled as revolute joints and twist angles are zero degrees.

Using kinematics with the Denavit-Hartenberg method of link transformation, the coordinate system of link *i* of an n-link open chain articulated rigid body with respect to *i*-1's coordinate system is found with a *A* matrix calculation. The *A*-matrix is defined as (Davidson 1993)

$$ {}_{i-1}^{\;\,i}A = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & \mathrm{a}_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & \mathrm{a}_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & \mathrm{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(Eq. 4.1)}$$

where θ, α, d, and a are the link parameters defined above.

The general equations of motion for a system of rigid bodies connected together with rotary joints and with control torque, ***T***, acting at each joint is (Koozekanani 1983) given by Equation 4.2. In what follows, bold type indicates vectors. Thus,

$$J(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} = \boldsymbol{A}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + B\boldsymbol{T} \qquad \text{(Eq. 4.2)}$$

where

J(θ) is the system inertia matrix,

A is a vector of centrifugal, Coriolis and gravitational forces,

B is the control distribution matrix,

$\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ are vectors of joint angles, joint rates, and joint accelerations, respectively, and ***T*** is the vector of joint torques.

Solving Equation 4.2 for ***T***,

$$\boldsymbol{T} = B^{-1}J(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} - B^{-1}A(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}). \qquad \text{(Eq. 4.3)}$$

The inverse dynamics problem solves for the joint torque, ***T***, given joint angles, rates and accelerations (Equation 4.3).

Conversely, the joint acceleration can be directly solved for with a given torque. This acceleration can be integrated once to solve for velocity and twice for position. Although this solution seems straightforward, the complexity for a multiple links model makes using it impractical (Koozekanani et al.1983). A recursive form of this equation was developed by Stepanenko and Vukobratovic (1976) for an inverse dynamic model. Koozekanani et al. (1983) applied this method to a direct dynamics model and used a single open serial chain model for postural dynamics as an example.

1. Recursive Inverse Dynamics Model

As stated earlier, inverse dynamics involves solving for forces and moments given the position, velocity and acceleration of the body. For an n-link open chain articulated planar mechanism as demonstrated by Koozekanani et al. (1983), this method can be solved recursively.

Accelerations of each link are calculated from the inboard to the outboard link using Equations 4.4 and 4.5. The results are:

$$\ddot{x}_i = \ddot{x}_{i-1} - (l_{i-1} - d_{i-1})(\ddot{\theta}_{i-1}\sin\theta_{i-1} + \dot{\theta}_{i-1}^2\cos\theta_{i-1}) - d_i(\ddot{\theta}_i\sin\theta_i + \dot{\theta}_i^2\cos\theta_i) \qquad \text{(Eq. 4.4)}$$

$$\ddot{z}_i = -\ddot{z}_{i-1} - (l_{i-1} - d_{i-1})(\ddot{\theta}_{i-1}\cos\theta_{i-1} - \dot{\theta}_{i-1}^2\sin\theta_{i-1}) - d_i(\ddot{\theta}_i\cos\theta_i - \dot{\theta}_i^2\sin\theta_i) \qquad \text{(Eq. 4.5)}$$

Forces and moments are then calculated from the outboard link to inboard using equations 4.6, 4.7, and 4.8. For an open chain, the end link is open and, therfore, has no forces or moments acting on its outboard end. Figure 4.1 shows the free-body diagram for one of these links.

$$\overset{+}{\rightarrow}\sum F_{x_i} = m_i\ddot{x}_i \quad \Rightarrow \quad F_{x_i} = F_{x_{i+1}} + m_i\ddot{x}_i \tag{Eq. 4.6}$$

$$+\downarrow\sum F_{z_i} = m_i\ddot{z}_i \quad \Rightarrow \quad F_{z_i} = F_{z_{i+1}} + m_i\ddot{z}_i - m_i g \tag{Eq. 4.7}$$

$$\oplus\sum T_i = J_i\ddot{\theta}_i \quad \Rightarrow \quad T_i = T_{i+1} - F_{x_{i+1}}(\,l_i - d_i\,)\sin\theta_i + F_{z_{i+1}}(\,l_i - d_i\,)\cos\theta_i + J_i\ddot{\theta}_i - F_{x_i} d_i \sin\theta_i + F_{z_i} d_i \cos\theta_i \tag{Eq. 4.8}$$

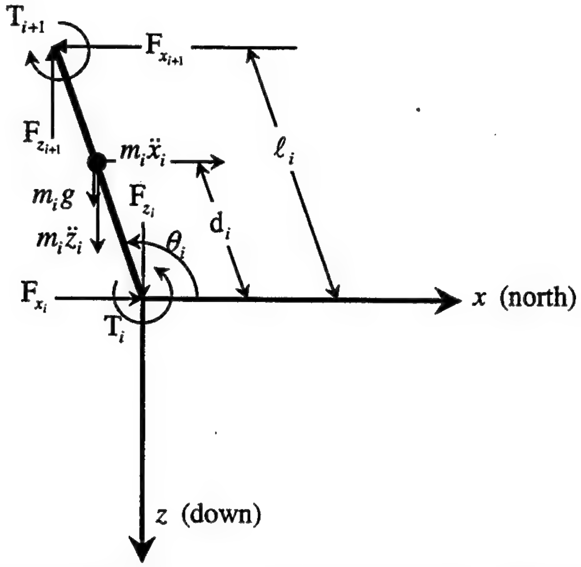

Figure 4.1. Free-body diagram for link i of an n-link serial open-chain articulaed planar mechanism

2. Recursive Direct Dynamics Model

Koozekanani et al. (1983) demonstrated the extension of the recursive inverse dynamics method to a recursive direct dynamics model for an n-link serial open-chain articulated planar mechanism on a postural control model. Solving Equation 4.2 for angular acceleration, the Lagrangian formulation of linkage dynamics of the form

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{J}^{-1}\boldsymbol{A} + \boldsymbol{J}^{-1}\boldsymbol{BT} \tag{Eq. 4.9}$$

will eventually be reduced to a simpler equation of the form

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{C}^{-1}(\boldsymbol{T} - \boldsymbol{T}_0)\,. \tag{Eq. 4.10}$$

First let Equation 4.3 be of the form,

$$\boldsymbol{T} = \boldsymbol{C}\ddot{\boldsymbol{\theta}} - \boldsymbol{D} \qquad \text{(Eq. 4.11)}$$

where $C = B^{-1}J = [C_1 C_2 \cdots C_n]$ and $\boldsymbol{D} = B^{-1}\boldsymbol{A}$. If each link has an acceleration of zero, $\ddot{\boldsymbol{\theta}} = (0,0,\cdots,0)^T$, then solving equations 4.4 and 4.5 from link 1 to link n, and Equations 4.6 through 4.8, from link n to link 1, the torque vector,$\boldsymbol{T}_0$ is determined. From Equation 4.11, $\boldsymbol{T}_0 = -\boldsymbol{D}$, and $\boldsymbol{T}_0$ is called the "equilibrium" torque vector of length n.

Next, step through each link with unit acceleration, or $\ddot{\boldsymbol{\theta}} = (1,0,\cdots,0)^T$. The torque vector solved here will be called unit acceleration torque, $\boldsymbol{T}_1$. Since, from Equation 4.11, $\boldsymbol{T}_1 = \boldsymbol{C}_1 - \boldsymbol{D} = \boldsymbol{C}_1 + \boldsymbol{T}_0$, then

$$\boldsymbol{C}_1 = \boldsymbol{T}_1 - \boldsymbol{T}_0, \qquad \text{(Eq. 4.12)}$$

where $\boldsymbol{C}_1$ is the first column of the C matrix in Equation 4.11, or generally,

$$\boldsymbol{C}_i = \boldsymbol{T}_i - \boldsymbol{T}_0 \qquad \text{(Eq. 4.13)}$$

for $\ddot{\boldsymbol{\theta}} = (0,0,\cdots,1,\cdots 0,0)^T$. Inverting this C matrix and solving for angular acceleration, Equation 4.10 results where $\boldsymbol{T}$ is any arbitrary torque. As stated above, the angular velocity and position can then be determined through numerical integration.

The arbitrary torque vector, $\boldsymbol{T}$, can be obtained from the data calculated using inverse dynamics equations and data from observations during a gait analysis laboratory experiment. Koozekanani et al. (1983) used linear feedback in the form of Equation 4.14 for the input torque vector to achieve postural control.

$$\boldsymbol{T} = -\boldsymbol{k}_{\theta}(\boldsymbol{\theta} - \boldsymbol{\theta}_0) - \boldsymbol{k}_{\dot{\theta}}(\dot{\boldsymbol{\theta}}) \qquad \text{(Eq. 4.14)}$$

where $\boldsymbol{k}_{\theta}$ and $\boldsymbol{k}_{\dot{\theta}}$ are the joint feedback gains, and θ_0 is the positional goal.

E. SUMMARY

In order to determine the forces and moments at each joint, kinematic data must be available and inverse dynamics equations used for the calculations. In the case of postural control, a recursive direct dynamics model can be solved by using an applied torque vector calculated from linear feedback. These techniques can be applied to a four-link open-chain serial articulated planar mechanism for the purpose of studying the initial impact phase of human gait.

V. FOUR-LINK HUMAN MODEL

A. INTRODUCTION

The analytic computer model used in this thesis contains a foot and an open chain planar serial linkage as shown in Figure 5.1. The foot is treated as a three-dimensional rigid body. The shank, thigh, and upper body are each three-dimensional links with revolute joints. (Only two-dimensional dynamics were used for the links.) The linkage is connected to the foot using a fourth massless link, "link0," that "transfers" forces, moments, position, velocities, and accelerations between the foot and the shank. Link0 has a twist angle of 90° in order to "twist" the inboard joint z-axis of revolution to an outboard link axis of rotation (ankle rotation) about the y-axis.

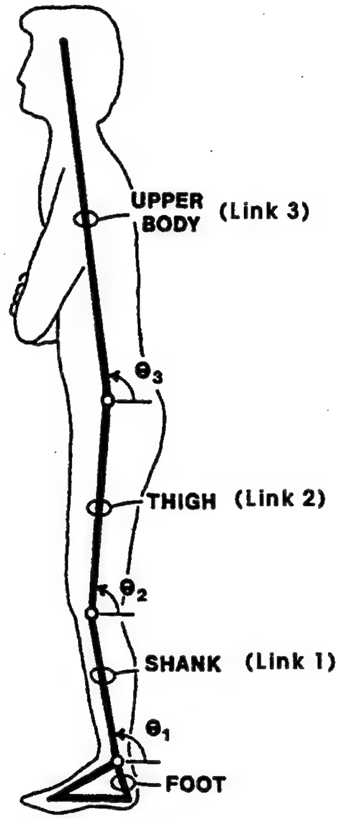

Figure 5.1. Four-mass sagittal plane model for the body from Koozekanani et al. (1980).

The foot is treated as a rigid body. The foot's position and Euler angles are used to properly position link0 while the foot's linear accelerations are used for the calculation of the linear acceleration of link1. The recursive Newton-Euler inverse-dynamics technique as described earlier is used to update the linear velocities of the remaining links. The forces and moments that act on the foot are then calculated, and the foot is updated. This cycle repeats until the loop ends.

This four-link inverted pendulum attached to a rigid body is an accurate representation of a human in a two-legged jump. The initial position of each link and body can be specified. The goal position can also be specified. With the correct initial conditions given, this model can satisfactorily represent the first few tenths of a second of heel strike during human gait. The importance of this phase of the gait with respect to knee injuries associated with articular cartilage changes that lead to chondromalacia and osteoarthritis has been shown earlier.

The coordinate system used is one with x-axis to the right (north), z-axis is down (down), and the y-axis is out of the paper (east) as shown in Figure 5.2.

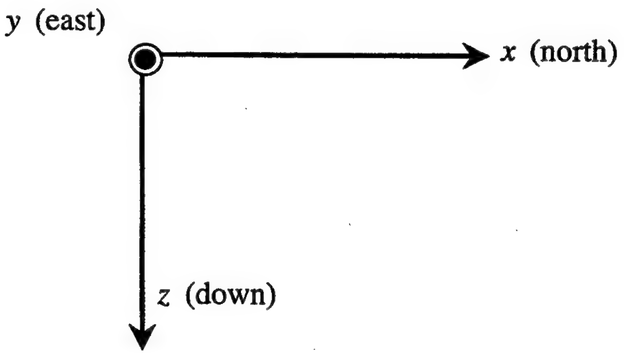

Figure 5.2. Coordinate system for four-link human

The anthropometric (dimension, weight, and moment of inertia) values of the foot and each link were obtained from Verstraete (1988), Knudson (1980), and Pandy (1987). Table 5.1 lists the values used in the model.

	Mass (slugs)	Length (ft)	Mass Moment of Inertia (slugs-ft^2)
Upper Body (link3)	3.75400	2.240	1.0000
Thigh (link2)	0.58740	1.447	0.0776
Shank (link1)	0.25176	1.550	0.0372
Foot	0.08112	0.492	0.0280

Table 5.1. Anthropometic data from Knudson (1980 pg. 21,34).

B. FOOT DYNAMICS

The foot's direct dynamics calculations are accomplished through the computer program titled "euler-angle-rigid-body" in Appendix A. This code is a three-dimensional rigid body direct dynamics solver for ANSI Common LISP. The initial values of all state variables were set in the initial form command line in each corresponding slot of the defining class function within the LISP program. The state vector (position, orientation, and body coordinate referenced linear and angular velocities) is sent to a Heun integration routine.

The forces and moments that are acting on the foot are calculated in the program file named "foot dynamics" (Appendix A). Figure 5.3 shows the free body diagram representing the foot. The quantities Fx_1, Fz_1, and M_1 are the forces and moments acting on the foot by the links. The forces acting at the heel and toe are reaction forces due to the spring and damping constants based on the mechanical properties of the ground. The recursive dynamics routine for the four inverted links use the earth-coordinate x-direction and z-direction accelerations calculated from the dynamics equation.

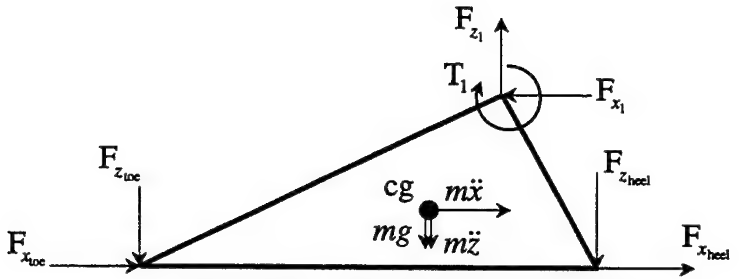

Figure 5.3. Foot free body diagram.

C. LINK DYNAMICS

The links' motions are calculated using the recursive Newton-Euler routine described in Chapter IV. After the accelerations of the foot are calculated for the given time step, these accelerations are used as inputs for link 1. Figure 5.4 shows a link free body diagram. The links' parameters for proper kinematic calculations are listed in Table 5.2.

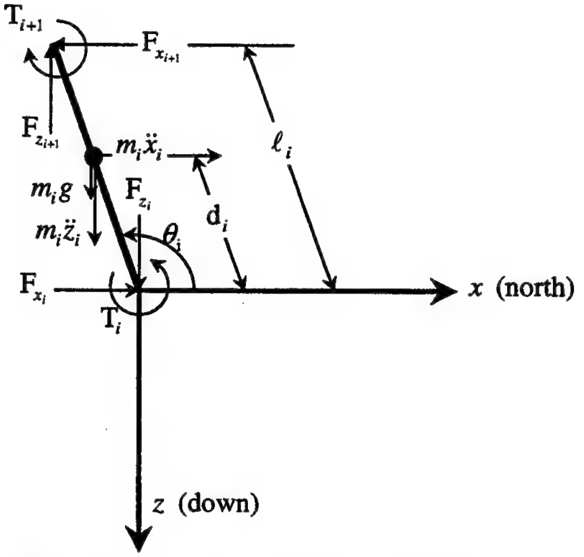

Figure 5.4. Link free body diagram.

Link	link length (a)	link twist (α)	link offset (d)	joint angle (θ)
Upper Body (link 3)	2.240	0°	0	θ_3
Thigh (link 2)	1.447	0°	0	θ_2
Shank (link 1)	1.550	0°	0	θ_1
Foot (link 0)	0.1905	-90°	-0.2286	0°

Table 5.2. Link Parameters.

The distance to the center of gravity (d_i) was assumed to be one-half of the link length. Knudson (1980) provides a much more accurate placement of the center of gravity for human limbs.

The foot is inboard to link 1, therefore, the foot's linear acceleration values are sent to link 1 for the calculations to be made of the rest of the links. Forces and moments that are calculated at the base of link 1 are forces and moments that act on the foot. The foot's dynamics encorperate these for the next time step's calculation of linear acceleration.

D. THE MISSING LINK

The most critical (and the most challenging) aspect of modeling the human gait using the methods cited above was connecting the foot to the shank link. This thesis arbitrarily chose to use the forces (Fx_1 and Fz_1) calculated during the equilibrium torque vector calculation to be the forces that act on the foot. The torque acting on the foot is the applied torque from the postural control feedback. The linear acceleration inputs for the inboard link of the shank are the linear accelerations of the foot. Both the foot and the inverted link system dynamics calculations were solved independently at each time step. The resulting dynamic model is not completely mathematically correct, but a crude approximation, as this method was chosen arbitrarily to test the model.

Two mathematically correct choices could have been made. The two methods, soft and hard constraints, would allow simultaneous solving of the link system and the foot. First, soft constraints could be used to connect the links to the foot. Alternatively, in order to fully utilize the capabilities of the Denavit-Hartenberg method, the foot can be simulated using hard constraints as a pair of sliding-link joints together with one rotary joint. The foot will then have 6 degrees of freedom, and the entire system can be solved simultaneously. Due to lack of time, this was not done in this thesis.

E. GROUND

The ground is modeled as a rigid body forming the x-y-plane at the origin of the z-axis. As any part of the foot begins to penetrate the "ground," a reaction force corresponding to its stiffness and damping acts on that part of the foot (Figure 5.5). Table 5.3 lists the stiffness of various surfaces (McMahon et al. 1979). The damping coefficients used in this thesis correspond to critical damping based on the stiffness.

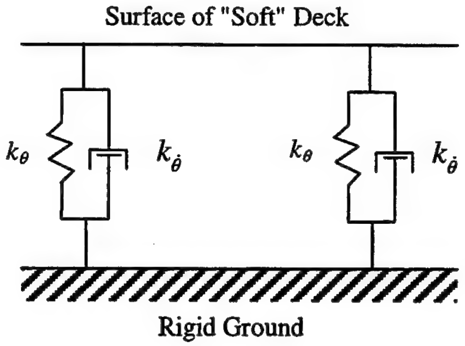

Figure 5.5. Surface stiffness and damping

Material	Stiffness (lbf/ft)
Concrete, asphalt	300,000+
Packed cinders	200,000
Board tracks	60,000
Experimental wooden track	133,333
Experimental wooden track	6857
Pillow tack at 1.67g	985

Table 5.3. Stiffness of various surfaces

F. ANSI COMMON LISP

The ANSI Common LISP (Graham 1996) code was used for all programming of the four-link-human model. LISP (LISt Processor) has the advantage of being an object oriented programming (OOP) language with the ability to test individual functions as they are defined. This feature permits a "bottom up" programming style which is very effective in complex system modeling. Use of OOP enables the programmer to write methods and functions that operate on and with the attributes of objects. Subclasses are able to inherit these attributes from their parent. Proponents argue that OOP is more analogous to the way a human thinks and therefore more intuitive than non-OOP. The results of this thesis demonstrate the versatility of OOP. The same program can demonstrate foot dynamics and postural control independently or together. The programmer only has to change a top-level function to use either or both.

Fundamental to understanding and programming in OOP is a class and object structure. Figure 5.6 shows the class and object structure of the four-link human. The four-link human was modified from Davidson's model of the aqua-robot (Davidson 1993).

G. SUMMARY

This chapter presents the analytic computer model that is used to solve for the initial impact impulse forces during human gait. Anthropometric data for the model and ground characteristics are detailed. A brief description of object-oriented programming and the structure of this model is also presented.

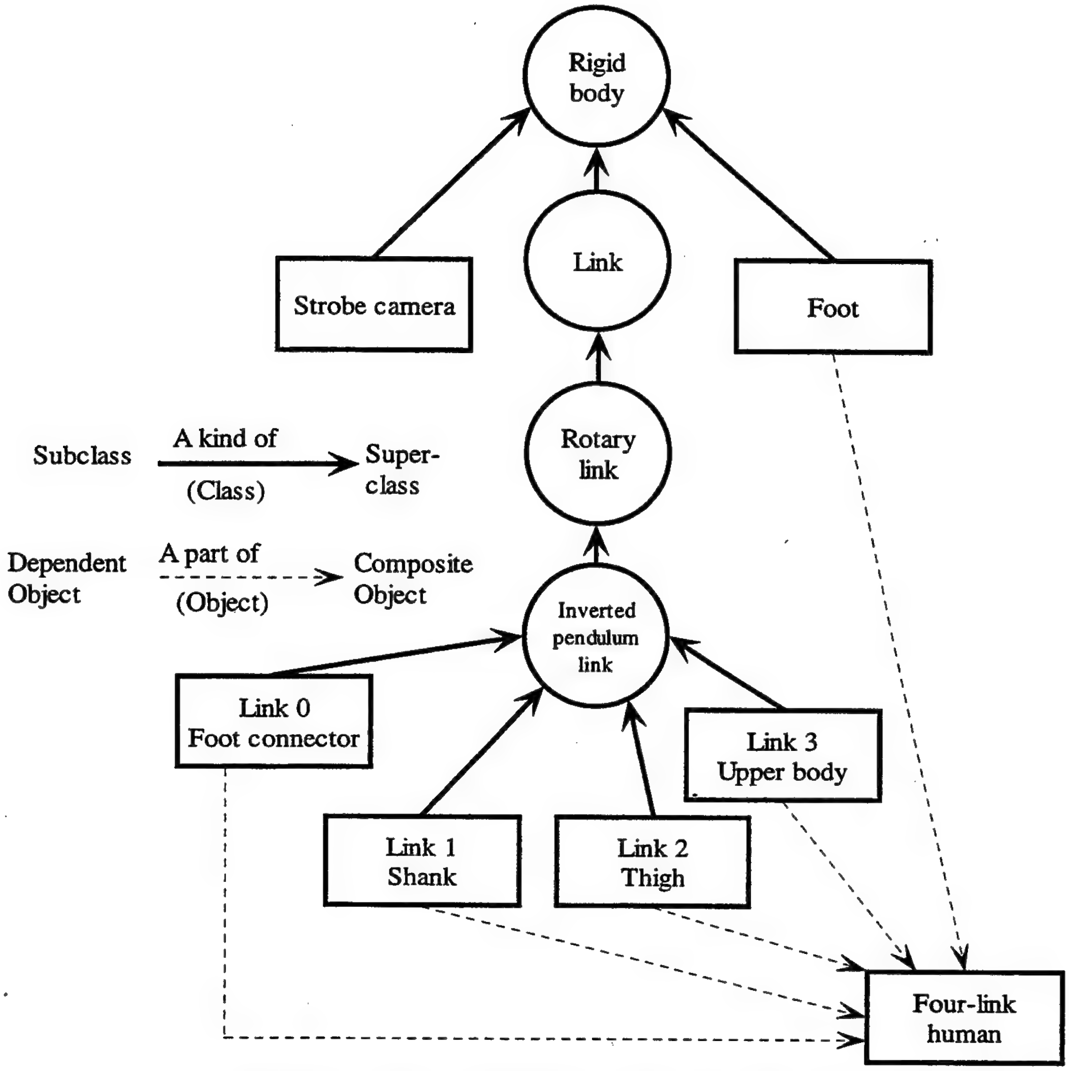

Figure 5.5. Class and Object structure of the four-link human.

VI. RESULTS

A. INTRODUCTION

The four-link human model was tested incrementally. One advantage of using LISP is that it allows the programmer to test each component separately and/or together. Only a different top-level function needs to be written for each case. This chapter is broken down into three sections as testing of the model was completed in stages. First, the foot was tested without the rest of the body linkage by setting and dropping it 1 cm with various initial conditions onto a surface. Next, the recursive direct dynamics calculations were tested using postural control. Finally, the two steps were brought together in the third section for testing the four-link human in the role of initial impact impulse force during human gait.

B. FOOT RIGID BODY DYNAMICS

1. Foot on "Soft" Surface

Figure 6.1 is the foot by itself sitting on the ground. Figure 6.2 is a vertical ground reaction force versus time plot as the foot settles into the ground. The foot was placed just touching the soft surface (z_{init} = -0.114286 ft.). The final position is slightly lower based on the weight of the foot and the spring constant of the ground (z_{final} = -0.11218 ft.). The center of gravity of the foot is 0.114286 ft above the bottom of the foot. The z-positions are negative because the ground is the coordinate axis with z-down being positive. For this "soft" surface, k_z = 1000 lbf/ft, and $k_{\dot{z}}$ = 20 lbf/ft/sec The foot dynamics routine was run for 0.2 seconds.

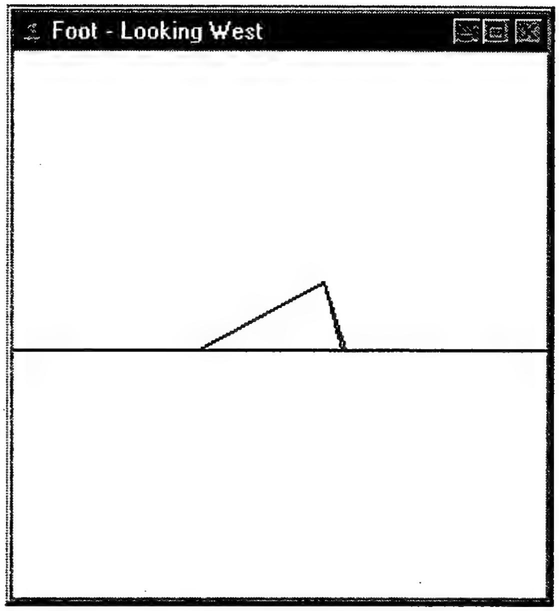

Figure 6.1. Foot on "Soft" surface.

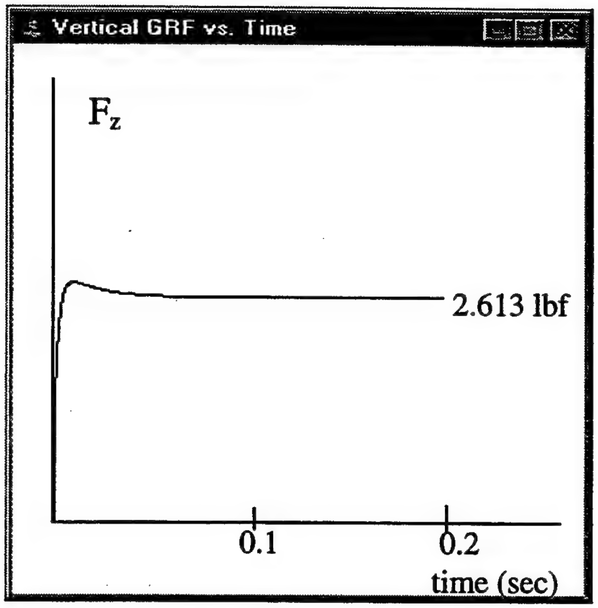

Figure 6.2. F_z versus time.

2. Foot Drop onto Surface

The next two figures (6.3 and 6.4) show the foot being dropped from the height of 0.0328 ft (1 cm) and the resulting vertical ground reaction force, respectively. For this "soft surface," $k_z = 1000$ lbf/ft, and $k_{\dot{z}} = 20$ lbf/ft/sec.

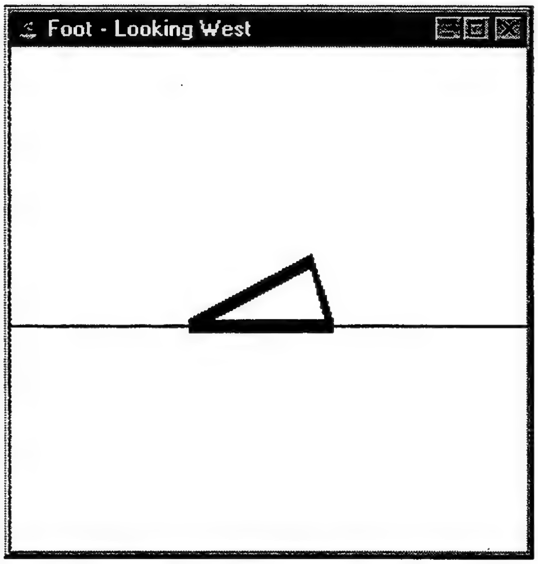

Figure 6.3. Foot drop from 1cm.

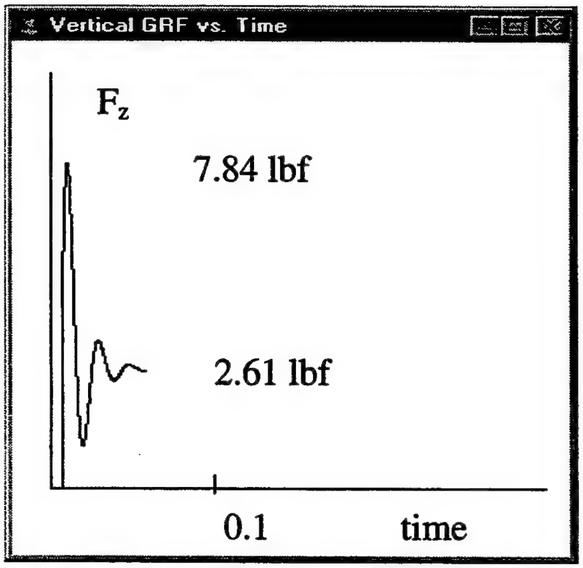

Figure 6.4. Vertical GRF vs. time.

3. Foot with Initial Angle Drop onto Surface

Figure 6.5 is the final step in testing the foot dynamics with an initial height and angle that models the foot just before it begins its free-fall. The heel is 1 cm above the ground. The angle of the foot is θ = -0.32175. The center of gravity is moved to properly position the heel (z_{cg} = - 0.2376). Figure 6.6 graphs the vertical ground reaction force versus time.

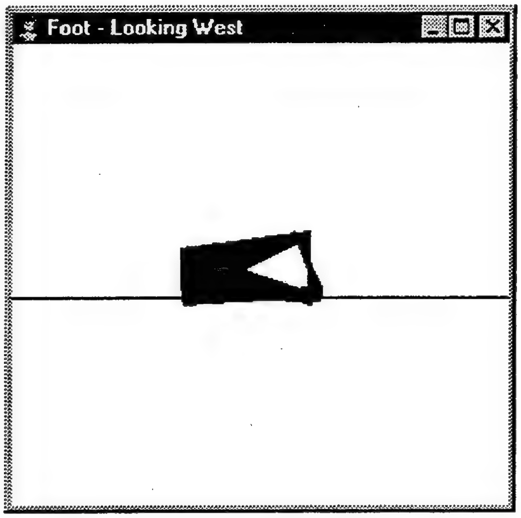

Figure 6.5. Foot fall with angle.

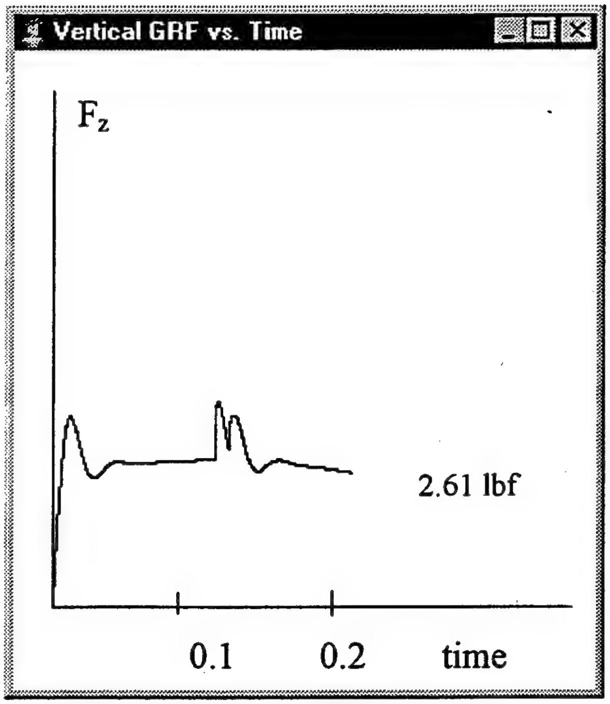

Figure 6.6. Vertical GRF vs. time.

C. RECURSIVE NEWTON-EULER DIRECT DYNAMICS

1. Postural Control with Foot Constrained

Figures 6.7 and 6.8 demonstrate the postural control ability of this model. The foot remains in its initial position and does not move. The model simulates Koozekanani's et al (1983) linear feedback postural control model. Stable linear feedback control gains are $\boldsymbol{k}_{\theta} = (500, 100, 35)^{\mathrm{T}}$ and $\boldsymbol{k}_{\dot{\theta}} = (500, 100, 30)^{\mathrm{T}}$. Both Figures show the goal position, the starting position, and the program stop position of the human model.

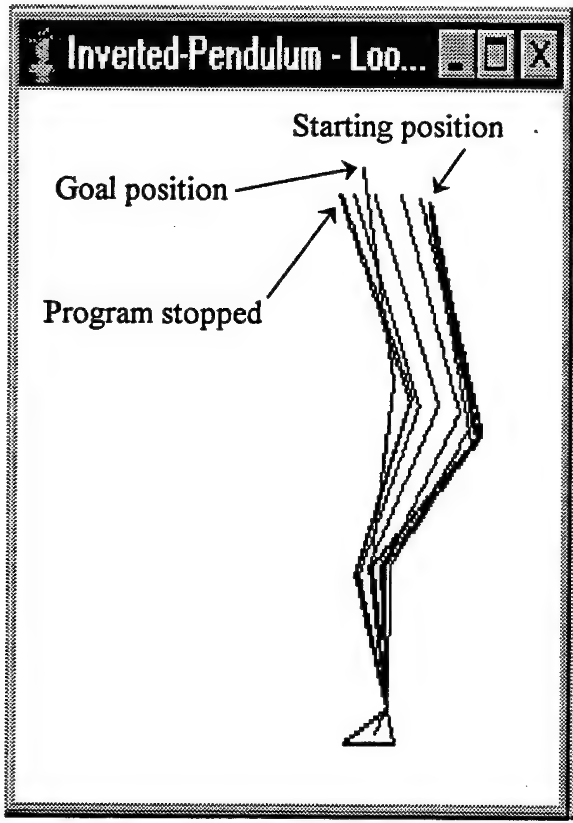

Figure 6.7. Postural Control

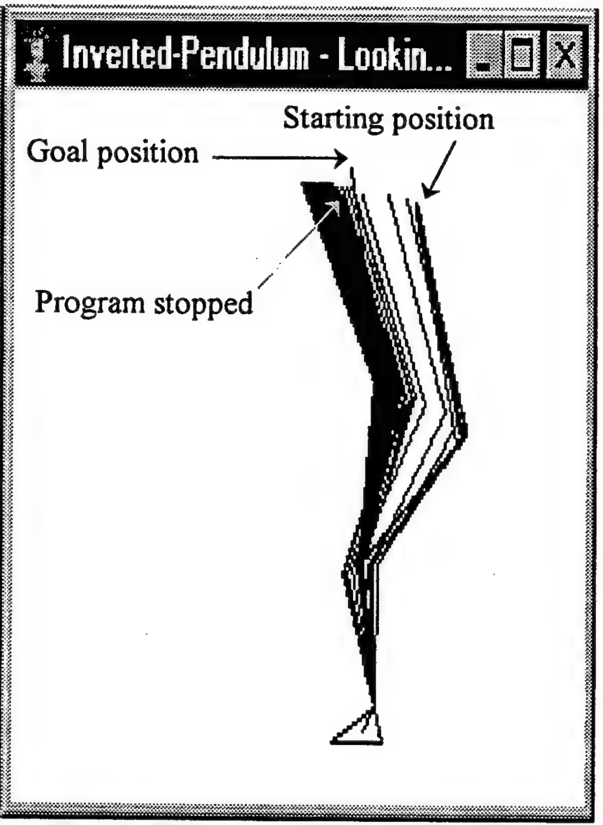

Figure 6.8. Postural Control Continued

2. Postural Control with Foot on "Soft" Surface

Figures 6.9 through 6.12 show the four-link human under postural control on a compliant surface for two different postures. Figures 6.9 and 6.10 are for a "bent" configuration, and Figures 6.11 and 6.12 are for a stiff or more upright stance. The gains were kept the same and the spring constant for the ground was $k_z = 1000$.

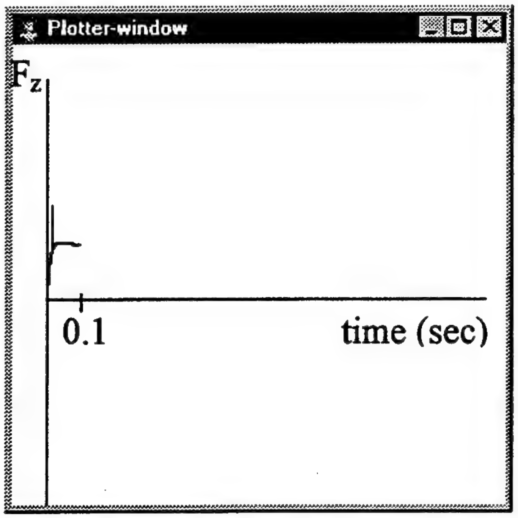

Figure 6.9. GRF vs. Time

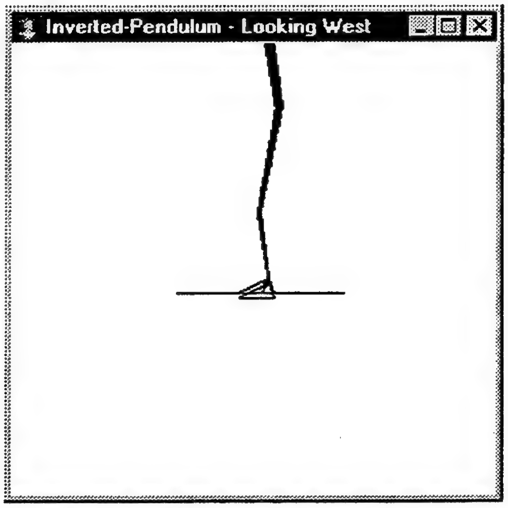

Figure 6.10. Postural control on soft surface for a bent stance

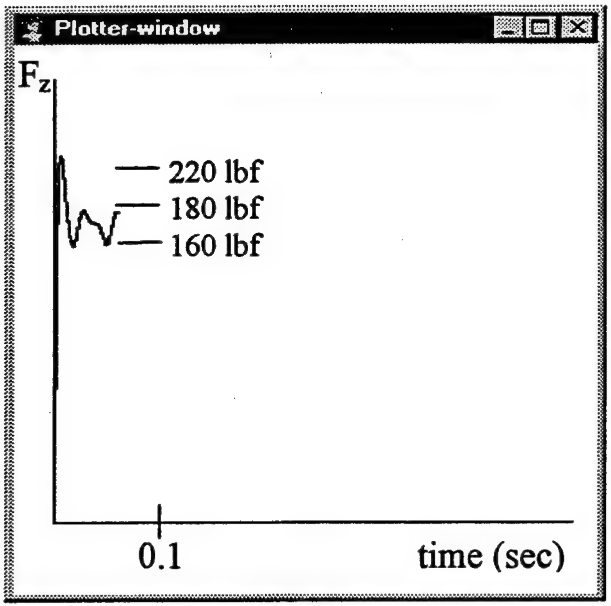

Figure 6.11. GRF vs. Time

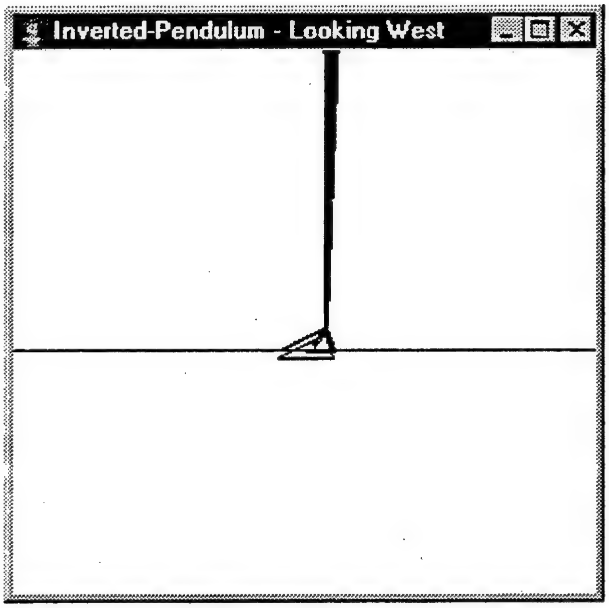

Figure 6.12. Postural control on soft surface for upright stance.

D. FOUR-LINK HUMAN DYNAMICS

1. Drop Human

Figure 6.13 shows the vertical ground reaction force of the four-link human "dropped" from just above the ground. No postural control was initiated during or after the drop.

2. Drop Test with Initial Conditions and Goal

Figure 6.14 shows the four-link human in the vertical position with a four-link human in the proper position for heel strike (Perry 1992).

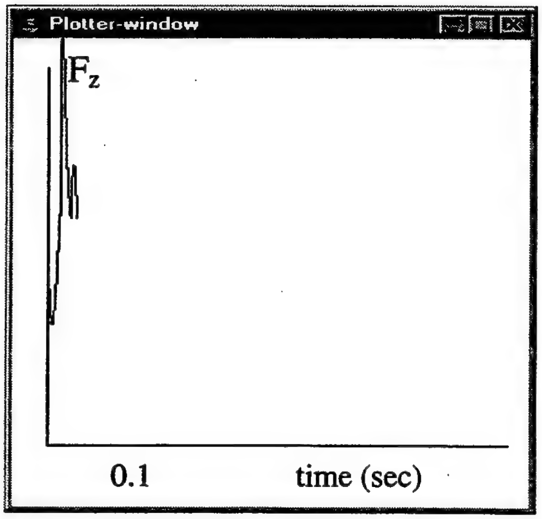

Figure 6.13 Vertical GRF vs. Time of dropped four-link human without postural control.

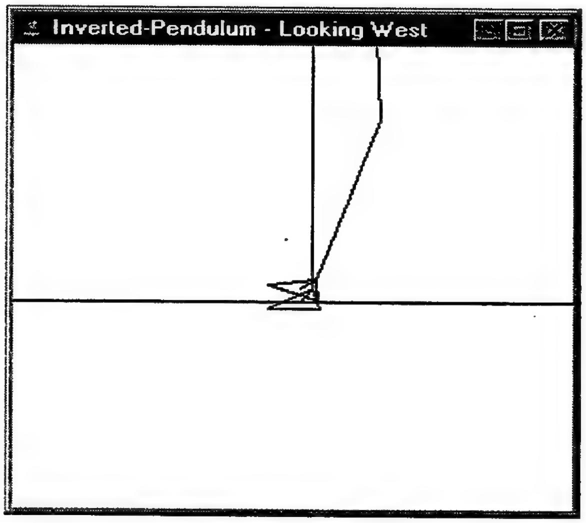

Figure 6.14. Proper heel strike position.

3. Heel Strike in Human Gait

Figure 6.15 shows an unstable first few milliseconds of a dropped four-link human with angled foot and postural control.

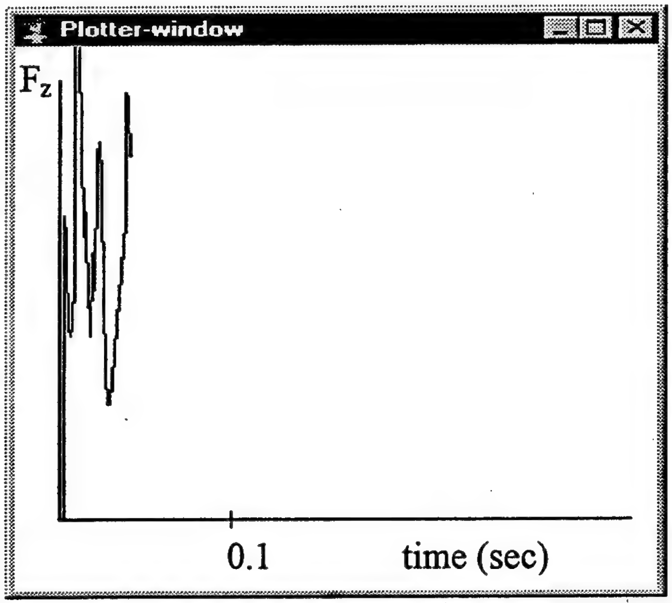

Figure 6.15. Vertical Ground Reaction Forces Versus Time.

E. SUMMARY

These results have shown the capabilities of the four-link human model and the ability of the programming language to test parts of or the whole model. The motion of the foot and links and the vertical ground reaction force plots seem to reflect a logical reality. However, before the model can be used to quantitatively evaluate deck surfaces, a more accurate hard or soft constraint joint must be used between the foot and the links. The model must then be independently verified through experimentation or another model.

THIS PAGE INTENTIONALLY LEFT BLANK.

VII. SUMMARY AND CONCLUSIONS

A. SUMMARY

The initial impact impulse force associated with human gait has been postulated to be a leading cause of disability of sailors. Placing a softer material between the foot and the hard steel deck should attenuate this impulsive force thereby reducing injuries, loss of training, and disabilities and the costs associated with them. The amount of attenuation or other changes to the characteristics of the initial impact impulse force must be quantified before decisions may be made as to the direction the U.S. Navy with respect to applying "softer" coating to deck as a preventative measure.

B. CONCLUSIONS

A tool has been developed to begin to quantify the ability to evaluate a softer material to reduce the impulse force of the initial contact or heel strike during human gait. However, as previously discussed, the connection between the foot and body used in this thesis is only an approximation used for code development, and is not mathematically correct. Consequently, simulation needs to be improved in this regard and then verified by independent simulation or experimental observation.

C. RECOMMENDATIONS

Throughout the development of this model, several assumptions were made and questions raised. The following recommendations are presented to improve the model, clarify the problem, and achieve an acceptable answer.

First, in order to improve the model, a mathematically correct mode for linking the foot to the body must be developed. Also, the model should be modified to include commanded velocities and initial condition velocities. The rest of the gait can then be modeled. The second leg, arms and head could also be added. The foot, being the interaction point between the body and the ground, can be simulated with better geometry, and possible two links itself with a flexor spring and damping. Add a spring for the tissue of the heel (D'Andrea et al. 1997) and a shoe (especially the spring and damping coefficients). Imagine a rigid foot hitting a steel deck. This model currently seems like a steel plate hitting the surface. The testing that can be accomplished with this program can be made more user friendly by building a graphical user interface (GUI) that would adjust initial positions, gains, spring constants, human properties, initial position, end position, etc. The ALLEGRO CL® "Dynamic Object-Oriented Programming System" (Franz Inc.) contains a useful tutorial to design, set up, and use a GUI for ANSI Common LISP programming.

Secondly, in order to clarify the problem, updated knee injuries statistics relating to articular cartilage damage within the Navy should be obtained in order to validate the cost associated with furthering this research. The Helmkamp and Coben (1985) study cited in Chapter II looked at sailors only after five to seven years of service. These types of injuries worsen with time. A study that looks at 15 to 20 years of service would also be helpful in determining the extent of these types of knee and joint problems. Additionally, because back pain has been tied to the initial impact impulse force (Collins and Whittle 1989a), the study should include back pain.

It would be useful to conduct a study equipping sailors with the low cost but accurate in-shoe force measuring device. Such measuring devices could be worn for extended periods of time in port and underway to determine the loading on the human body during shipboard life. In such a study, special emphasis should be placed on walking through hatches, transitioning off ladders, and climbing ladders.

Finally, in order to validate the model, a partnership can be made with an existing gait analysis laboratory. It would be especially valuable to put a hatch on a walkway equipped with a force plate and measure the ground reaction forces felt when a person steps through a smaller opening in which he or she must duck. Such motion requires a greater free fall and a more violent collision with the ground. Additionally, a ladder should be constructed on the runway. The ladder should be positioned so that the foot coming off of the ladder lands on the force plate (possibly two force plates, one for each foot).

THIS PAGE INTENTIONALLY LEFT BLANK.

APPENDIX A: ANSI COMMON LISP CODE

```
; ****************************************************************************
; File      :robot-kinematics.cl
; Author    :Dr. R. McGhee
;           :Naval Postgraduate School
;           :Monterey, CA 93943
; Summary   :Matrix and Vector Operations
; ****************************************************************************

(defun transpose (matrix)           ;A matrix is a list of row vectors.
  (cond ((null (cdr matrix)) (mapcar 'list (car matrix)))
        (t (mapcar 'cons (car matrix) (transpose (cdr matrix))))))

(defun dot-product (vector-1 vector-2) ;A vector is a list of numerical atoms.
  (apply '+ (mapcar '* vector-1 vector-2)))

(defun cross-product (vector-1 vector-2)
  (let ((x1 (first vector-1)) (y1 (second vector-1)) (z1 (third vector-1))
        (x2 (first vector-2)) (y2 (second vector-2)) (z2 (third vector-2)))
       (list (- (* y1 z2) (* y2 z1)) (- (* x2 z1) (* x1 z2))
             (- (* x1 y2) (* x2 y1)))))

(defun vector-magnitude (vector) (sqrt (dot-product vector vector)))

(defun post-multiply (matrix vector)
  (cond ((null (rest matrix)) (list (dot-product (first matrix) vector)))
        (t (cons (dot-product (first matrix) vector)
                 (post-multiply (rest matrix) vector)))))

(defun pre-multiply (vector matrix)
  (post-multiply (transpose matrix) vector))

(defun matrix-multiply (matrix1 matrix2)
  (cond ((null (rest matrix1)) (list (pre-multiply (first matrix1) matrix2)))
        (t (cons (pre-multiply (first matrix1) matrix2)
                 (matrix-multiply (rest matrix1) matrix2)))))

(defun chain-multiply (L)      ;L is a list of names of conformable matrices.
  (cond ((null (cddr L)) (matrix-multiply (eval (car L)) (eval (cadr L))))
        (t (matrix-multiply (eval (car L)) (chain-multiply (cdr L))))))

(defun cycle-left (matrix) (mapcar 'row-cycle-left matrix))

(defun row-cycle-left (row) (append (cdr row) (list (car row))))

(defun cycle-up (matrix) (append (cdr matrix) (list (car matrix))))

(defun unit-vector (one-column length)   ;Column count starts at 1.
  (do ((n length (1- n))
       (vector nil (cons (cond ((= one-column n) 1) (t 0)) vector)))
      ((zerop n) vector)))

(defun unit-matrix (size)
  (do ((row-number size (1- row-number))
       (I nil (cons (unit-vector row-number size) I)))
      ((zerop row-number) I)))

(defun concat-matrix (matrix1 matrix2)
  (if matrix1 (cons (append (first matrix1) (first matrix2))
                    (concat-matrix (rest matrix1) (rest matrix2)))))

(defun augment (matrix)
  (concat-matrix matrix (unit-matrix (length matrix))))

(defun normalize-row (row) (scalar-multiply (/ 1.0 (first row)) row))

(defun scalar-multiply (scalar vector)
  (cond ((null vector) nil)
        (t (cons (* scalar (first vector))
                 (scalar-multiply scalar (rest vector))))))

(defun solve-first-column (matrix)  ;Reduces first column to (1 0 ... 0).
  (do* ((remaining-row-list matrix (rest remaining-row-list))
        (first-row (normalize-row (first matrix)))
        (answer (list first-row)
                (cons (vector-add (first remaining-row-list)
                                  (scalar-multiply (- (caar remaining-row-list))
                                                    first-row))
                      answer)))
       ((null (rest remaining-row-list)) (reverse answer))))

(defun vector-add (vector-1 vector-2) (mapcar '+ vector-1 vector-2))

(defun vector-subtract (vector-1 vector-2) (mapcar '- vector-1 vector-2))

(defun matrix-subtract (matrix-1 matrix-2)
  (mapcar #'vector-subtract matrix-1 matrix-2))

(defun subtract-unit-matrix (square-matrix)
  (matrix-subtract square-matrix (unit-matrix (length square-matrix))))

(defun sum-of-elements-squared (matrix)
  (apply '+ (mapcar #'dot-product matrix matrix)))

(defun rms-inverse-error-metric (matrix approximate-inverse-matrix)
  (let* ((M matrix) (M-inv approximate-inverse-matrix) (n (length M))
         (error-matrix (subtract-unit-matrix (matrix-multiply M M-inv)))
         (S (sum-of-elements-squared error-matrix)))
        (/ (sqrt S) n)))

(defun first-square (matrix)  ;Returns leftmost square matrix from argument.
  (do ((size (length matrix))
       (remainder matrix (rest remainder))
       (answer nil (cons (firstn size (first remainder)) answer)))
      ((null remainder) (reverse answer))))

(defun firstn (n list)
  (cond ((zerop n) nil)
         (t (cons (first list) (firstn (1- n) (rest list))))))

(defun max-car-firstn (n list)
  (append (max-car-first (firstn n list)) (nthcdr n list)))

(defun matrix-inverse (matrix)
  (do* ((M (max-car-first (augment matrix))
           (max-car-firstn n (cycle-left (cycle-up M))))
        (n (1- (length matrix)) (1- n))
        (exit-flag (= 0 (caar M)) (= 0 (caar M)))) ;Prevents division by zero.
       ((or (minusp n) exit-flag) (if (not exit-flag) (first-square M)))
       (setf M (solve-first-column M))))

(defun max-car-first (matrix)      ;This function finds row with largest first
  (cond ((null (cdr matrix)) matrix)    ;element and moves it to top of matrix.
        (t (if (> (abs (caar matrix))
                  (abs (caar (max-car-first (cdr matrix))))) matrix
               (append (max-car-first (cdr matrix)) (list (car matrix)))))))
```

```
(defun dh-matrix (rotate twist length translate)
  (let ((cosrotate (cos rotate)) (sinrotate (sin rotate))
        (costwist (cos twist)) (sintwist (sin twist)))
       (list (list cosrotate (- (* costwist sinrotate))
                   (* sintwist sinrotate) (* length cosrotate))
             (list sinrotate (* costwist cosrotate)
                   (- (* sintwist cosrotate)) (* length sinrotate))
             (list 0. sintwist costwist translate)
             (list 0. 0. 0. 1.))))

(defun homogeneous-transform (orientation position)
  (let* ((roll (first orientation)) (elevation (second orientation))
         (azimuth (third orientation)) (x (first position))
         (y (second position)) (z (third position))
         (spsi (sin azimuth)) (cpsi (cos azimuth)) (sth (sin elevation))
         (cth (cos elevation)) (sphi (sin roll)) (cphi (cos roll)))
        (list (list (* cpsi cth) (- (* cpsi sth sphi) (* spsi cphi))
                    (+ (* cpsi sth cphi) (* spsi sphi)) x)
              (list (* spsi cth) (+ (* cpsi cphi) (* spsi sth sphi))
                    (- (* spsi sth cphi) (* cpsi sphi)) y)
              (list (- sth) (* cth sphi) (* cth cphi) z)
              (list 0. 0. 0. 1.))))

(defun inverse-H (H)         ;H is a 4x4 homogeneous transformation matrix.
  (let* ((minus-P (list (- (fourth (first H)))
                        (- (fourth (second H)))
                        (- (fourth (third H)))))
         (inverse-R (transpose (first-square (reverse (rest (reverse H))))))
         (inverse-P (post-multiply inverse-R minus-P)))
        (append (concat-matrix inverse-R (transpose (list inverse-P)))
                (list (list 0 0 0 1)))))

(defun rotation-matrix (euler-angles)
  (let* ((roll (first euler-angles)) (elevation (second euler-angles))
         (azimuth (third euler-angles))
         (spsi (sin azimuth)) (cpsi (cos azimuth)) (sth (sin elevation))
         (cth (cos elevation)) (sphi (sin roll)) (cphi (cos roll)))
        (list (list (* cpsi cth) (- (* cpsi sth sphi) (* spsi cphi))
                    (+ (* cpsi sth cphi) (* spsi sphi)))
              (list (* spsi cth) (+ (* cpsi cphi) (* spsi sth sphi))
                    (- (* spsi sth cphi) (* cpsi sphi)))
              (list (- sth) (* cth sphi) (* cth cphi)))))

(defun body-rate-to-euler-rate-matrix (euler-angles)
  (let* ((roll (first euler-angles)) (elevation (second euler-angles))
         (sth (sin elevation)) (cth (cos elevation)) (tth (tan elevation))
         (sphi (sin roll)) (cphi (cos roll)))
       (list (list 1 (* tth sphi) (* tth cphi))
             (list 0 cphi (- sphi))
             (list 0 (/ sphi cth) (/ cphi cth)))))

(defun rad-to-deg (angle) (* 57.29577951308232 angle))

(defun deg-to-rad (angle) (* 0.017453292519943295 angle))
```

```
; *****************************************************************************
; File      :numerical-integration.cl
; Author    :Dr. R. McGhee
;           :Naval Postgraduate School
;           :Monterey, CA 93943
; Summary   :Euler, Heun, and Runga Kutta 45 Integration
; *****************************************************************************

(defun euler-step (x time time-step) ;x is list representation of state vector.
  (vector-add x (euler-increment x time time-step)))

(defun euler-increment (x time time-step)
  (scalar-multiply time-step (derivative x time)))

(defun heun-step (x time time-step)
  (vector-add x (scalar-multiply (* time-step .5)
    (vector-add (derivative x time) (derivative (euler-step x time time-step)
                                                (+ time time-step))))))

(defun RK4-step (x time time-step)
  (let* ((dt (* time-step .5)) (delta-x0 (euler-increment x time dt))
         (delta-x1 (euler-increment (vector-add x delta-x0) (+ time dt) dt))
         (delta-x2 (euler-increment (vector-add x delta-x1) (+ time dt) dt))
         (2dx2 (scalar-multiply 2 delta-x2))
         (delta-x3 (euler-increment (vector-add x 2dx2) (+ time time-step) dt))
         (sum (RK4-sum delta-x0 delta-x1 delta-x2 delta-x3))
         (RK4-increment (scalar-multiply 1/3 sum)))
        (vector-add x RK4-increment)))

(defun integration-step (method x time  time-step)
  (cond ((equal method 'euler-step) (euler-step x time time-step))
        ((equal method 'heun-step) (heun-step x time time-step))
        ((equal method 'RK4-step) (RK4-step x time time-step))))

(defun RK4-sum (delta-1 delta-2 delta-3 delta-4)
  (let* ((d1 delta-1) (d2 delta-2) (d3 delta-3) (d4 delta-4)
         (sum1 (vector-add d1 d4)) (sum2 (vector-add d2 d3)))
        (vector-add sum1 (scalar-multiply 2 sum2))))
```

```
; ****************************************************************************
; File       :plotter.cl
; Author     :David A. Bretz
;            :Naval Postgraduate School
;            :Monterey, CA 93943
; Summary    :Draws line in plot window instance form old-value to new-value
;            :It is not user friendly, ie., it must be changes manually for
;            :different x (horizontal) and y (vertical) values.
;            :All functions in this file are taken from Dr. R. McGhee's
;            :Strobe-Camera
; ****************************************************************************

(defclass plotter ()
  ((plotter-window
    :accessor plotter-window
    :initform (open-stream 'bitmap-window
                  *lisp-main-window*
                  :output
                  :window-exterior
                      (make-box 720 300 1020 650)
                  :title "Plotter-window"))
   (old-value
    :initform '(0 0)
    :initarg :old-value
    :accessor old-value)
   (new-value
    :initform '(0 0)
    :initarg :new-value
    :accessor new-value)))

(defmethod plot ((plotter plotter) x-data y-data)
  (setf (new-value plotter) (list x-data y-data))
  (draw-line-in-window (plotter-window plotter)
                       10
                       (old-value plotter) (new-value plotter))
  (setf (old-value plotter) (list x-data y-data)))


(defun draw-coordinate-axes (window)
  (draw-line window (make-position  20 280)   ; was 20 150
                    (make-position  280 280)) ; was 280 150
  (draw-line window (make-position 20 20)
                    (make-position 20 280)))

(defun scale-point-coordinates (x-y-list enlargement-factor)
  (let ((x (first x-y-list)) (y (second x-y-list)))
       (list (+ 20 (round (* enlargement-factor x 20)))
             (+ 280 (round (* (- y))))))) ;was + 150

(defun draw-line-in-window (window enlargement-factor start end)
  (let ((scaled-start (scale-point-coordinates start enlargement-factor))
        (scaled-end (scale-point-coordinates end enlargement-factor)))
       (draw-line window
         (make-position (first scaled-start) (second scaled-start))
         (make-position (first scaled-end)  (second scaled-end)))))
```

```
; ***************************************************************************
; File             :euler-angle-rigid-body.cl
; Author           :Dr. R. McGhee
;                  :Naval Postgraduate School
;                  :Monterey, CA 93943
; Summary          :Rigid body class written and implemented by Dr. R. McGhee.
; ***************************************************************************

(defclass rigid-body ()
  ((euler-angles      ;The vector (psi theta phi) = (roll elevation azimuth).
    :initform '(0 0 0)
    :initarg :euler-angles
    :accessor euler-angles)
   (cg-position                ;The vector (xe ye ze) = (north east down).
    :initform '(0 0 0)
    :initarg :cg-postion
    :accessor cg-position)
   (body-coord-angular-velocity  ;The vector (p q r) = (roll-rate pitch-rate
    :initform '(0 0 0)          ;                              yaw-rate)
    :initarg :body-coord-angular-velocity
    :accessor body-coord-angular-velocity)
   (body-coord-linear-velocity  ;The vector (u v w) in body coordinates.
    :initform '(0 0 0)
    :initarg :body-coord-linear-velocity
    :accessor body-coord-linear-velocity)
   (moment-of-inertia-matrix  ;Moments of inertia about center of gravity.
    :initform (unit-matrix 3)
    :initarg :moment-of-inertia-matrix
    :accessor moment-of-inertia-matrix)
   (mass
    :initform 1
    :initarg :mass
    :accessor mass)
   (node-list    ;(x y z 1) in body coord for each node. Starts with (0 0 0 1).
    :initform '((0 0 0 1) (0 0 0 1) (0 0 0 1))
    :initarg :node-list
    :accessor node-list)
   (polygon-list
    :initform '(1 2)
    :initarg :polygon-list
    :accessor polygon-list)
   (transformed-node-list  ;(x y z 1) in earth coord for each node in node-list.
    :accessor transformed-node-list)
   (H-matrix
    :initform (unit-matrix 4)
    :accessor H-matrix)
   (clock-tick-count
    :accessor clock-tick-count)
   (time-stamp
    :initform 0
    :accessor time-stamp)))

(defmethod initialize ((body rigid-body))
  (setf (transformed-node-list body) (node-list body)
        (clock-tick-count body) (get-internal-real-time)
        *inverse-mass* (/ 1 (mass body))
        *I* (moment-of-inertia-matrix body)
        *I-inv* (matrix-inverse *I*)))
```

```
(defmethod move ((body rigid-body) orientation position)
  (setf (H-matrix body) (homogeneous-transform orientation position)
        (euler-angles body) orientation (cg-position body) position)
  (transform-node-list body))

(defmethod get-delta-t ((body rigid-body) read-clock-flag default-value)
  (if read-clock-flag
      (let* ((old-time (clock-tick-count body))
            (new-time (setf (clock-tick-count body) (get-internal-real-time))))
           (/ (- new-time old-time) 1000))
     default-value))

(defmethod update-rigid-body ((body rigid-body) real-time-flag default-dt)
  (let* ((delta-t (get-delta-t body real-time-flag default-dt))
         (time (time-stamp body))
         (pqr (body-coord-angular-velocity body))
         (uvw (body-coord-linear-velocity body))
         (x (append (cg-position body) (euler-angles body) uvw pqr))
         (new-x (heun-step x time delta-t)))
        (store-state body new-x)
        (setf (H-matrix body)
              (homogeneous-transform (euler-angles body) (cg-position body)))
        (transform-node-list body)
        (setf (time-stamp body) (+ time delta-t))))

(defun derivative (state time)
  (let* ((fm (forces-moments state time))
         (f (first-three fm))
         (fm (nthcdr 3 fm))
         (moments (first-three fm))
         ;(f (forces state time)) (moments (moments state time))
         (location (first-three state)) (state (nthcdr 3 state))
         (angles (first-three state)) (state (nthcdr 3 state))
         (uvw (first-three state)) (state (nthcdr 3 state)) (pqr state)
         (R (rotation-matrix angles)) (v-earth (post-multiply R uvw))
         (M (body-rate-to-euler-rate-matrix angles))
         (euler-angle-rates (post-multiply M pqr))
         (g-body (post-multiply (transpose R) *gravity*))
         (negative-pqr (scalar-multiply -1 pqr))
         (f/m (scalar-multiply *inverse-mass* f))
         (coreolis (cross-product negative-pqr v-earth))
         (uvw-dot (vector-add f/m (vector-add g-body coreolis)))
         (H (post-multiply *I* pqr))
         (pqr-dot (post-multiply *I-inv* (vector-subtract moments (cross-product
pqr H)))))
   (setf (output-vector-to-link1 (foot human-1)) (append f/m pqr-dot euler-
angle-rates angles))
   (append v-earth euler-angle-rates uvw-dot pqr-dot)))

(defmethod store-state ((body rigid-body) state)
  (let* ((location (first-three state)) (state (nthcdr 3 state))
         (angles (first-three state)) (state (nthcdr 3 state))
         (uvw (first-three state)) (state (nthcdr 3 state)) (pqr state))
        (setf (euler-angles body) angles (cg-position body) location
              (body-coord-angular-velocity body) pqr
              (body-coord-linear-velocity body) uvw)))
```

```
(defun first-three (list) (list (first list) (second list) (third list)))

(defmethod transform-node-list ((body rigid-body))
  (setf (transformed-node-list body)
    (mapcar #'(lambda (node-location)
                (post-multiply (H-matrix body) node-location))
            (node-list body))))

(defconstant *gravity* '(0 0 32.2185))

;(defun forces (state time) (scalar-multiply -1 *gravity*))

(defun moments (state time) '(0 0 0))

(defun test ()
  (setf airplane-1 (make-instance 'rigid-body))
  (setf camera-1 (make-instance 'strobe-camera))
  (move camera-1 (list 0 (- (/ pi 2)) 0) '(0 0 -30))
  (initialize airplane-1)
  (take-picture camera-1 airplane-1)
  (dotimes (i 20 'done) (update-rigid-body airplane-1 nil .1))
  (take-picture camera-1 airplane-1)
  (time-stamp airplane-1))

(defun repeat ()
  (initialize airplane-1)
  (dotimes (i 10 'done) (update-rigid-body airplane-1 nil .1))
  (take-picture camera-1 airplane-1)
  (time-stamp airplane-1))
```

```
; ****************************************************************************
; File        :PC-camera.cl
; Author      :Dr. R. McGhee
;             :Naval Postgraduate School
;             :Monterey, CA 93943
; Summary     :"Takes picture" of a rigid body
; ****************************************************************************

(defclass strobe-camera (rigid-body)
  ((focal-length
    :accessor focal-length
    :initform 1)
   (camera-window
    :accessor camera-window
    :initform(open-stream 'bitmap-window
                  *lisp-main-window*
                  :output
                  :window-exterior
                      (make-box 720 0 1020 300)
                  :title "Inverted-Pendulum - Looking West"))
   (H-matrix     ;The first vector (psi theta phi) = (roll elevation azimuth).
    :accessor H-matrix    ;The second vector (xe ye ze) = (north east down).
    :initform (homogeneous-transform '(0 0 -1.57) '(0 3 0)))
   (inverse-H-matrix
    :accessor inverse-H-matrix
    :initform (inverse-H (homogeneous-transform '(0 0 -1.57) '(0 3 0))))
   (enlargement-factor
    :accessor enlargement-factor
    :initform 10)))

(defmethod move-body ((camera strobe-camera) orientation position)
  (setf (H-matrix camera) (homogeneous-transform orientation position))
  (setf (inverse-H-matrix camera) (inverse-H (H-matrix camera))))

(defmethod take-picture ((camera strobe-camera) (body rigid-body))
  (let ((camera-space-node-list (mapcar #'(lambda (node-location)
         (post-multiply (inverse-H-matrix camera) node-location))
                        (transformed-node-list body))))
    (dolist (polygon (polygon-list body))
      (clip-and-draw-polygon camera polygon camera-space-node-list))))

(defclass still-camera (strobe-camera) ())

(defmethod take-picture ((camera still-camera) (body rigid-body))
  (clear-page (camera-window camera))
  (let ((camera-space-node-list (mapcar #'(lambda (node-location)
         (post-multiply (inverse-H-matrix camera) node-location))
                        (transformed-node-list body))))
    (dolist (polygon (polygon-list body))
      (clip-and-draw-polygon camera polygon camera-space-node-list))))

(defmethod clip-and-draw-polygon
  ((camera strobe-camera) polygon node-coord-list)
  (do* ((initial-point (nth (first polygon) node-coord-list))
        (from-point initial-point to-point)
        (remaining-nodes (rest polygon) (rest remaining-nodes))
        (to-point (nth (first remaining-nodes) node-coord-list)
                  (if (not (null (first remaining-nodes)))
                      (nth (first remaining-nodes) node-coord-list))))
       ((null to-point)
              (draw-clipped-projection camera from-point initial-point))
       (draw-clipped-projection camera from-point to-point)))

(defmethod draw-clipped-projection ((camera strobe-camera) from-point to-point)
  (cond ((and (<= (first from-point) (focal-length camera))
              (<= (first to-point) (focal-length camera))) nil)
        ((<= (first from-point) (focal-length camera))
         (draw-line-in-camera-window camera
           (perspective-transform camera (from-clip camera from-point to-point))
           (perspective-transform camera to-point)))
        ((<= (first to-point) (focal-length camera))
         (draw-line-in-camera-window camera
           (perspective-transform camera from-point)
           (perspective-transform camera (to-clip camera from-point to-point))))
        (t (draw-line-in-camera-window camera
           (perspective-transform camera from-point)
           (perspective-transform camera to-point)))))

(defmethod from-clip ((camera strobe-camera) from-point to-point)
  (let ((scale-factor (/ (- (focal-length camera) (first from-point))
                         (- (first to-point) (first from-point)))))
       (list (+ (first from-point)
                (* scale-factor (- (first to-point) (first from-point))))
             (+ (second from-point)
                (* scale-factor (- (second to-point) (second from-point))))
             (+ (third from-point)
                (* scale-factor (- (third to-point) (third from-point)))) 1)))

(defmethod to-clip ((camera strobe-camera) from-point to-point)
  (from-clip camera to-point from-point))

(defmethod draw-line-in-camera-window ((camera strobe-camera) start end)
  (draw-line (camera-window camera)
      (make-position (first start) (second start))
      (make-position (first end) (second end))))

(defmethod perspective-transform ((camera strobe-camera) point-in-camera-space)
  (let* ((enlargement-factor (enlargement-factor camera))
         (focal-length (focal-length camera))
         (x (first point-in-camera-space))  ;x axis is along optical axis
         (y (second point-in-camera-space)) ;y is out right side of camera
         (z (third point-in-camera-space))) ;z is out bottom of camera
        (list (+ (round (* enlargement-factor (/ (* focal-length y) x)))
                 150)                       ;to right in camera window
              (+ 150 (round (* enlargement-factor (/ (* focal-length z) x))
                                 )))))      ;up in camera window
```

```
; ****************************************************************************
; File          :link.cl
; Author        :Dr. R. McGhee
;               :Naval Postgraduate School
;               :Monterey, CA 93943
; Modified by   :David A. Bretz
; Summary       :Defines the classes of a rigid body link, rotary link, and
;               :  inverted pendulum link
; ****************************************************************************

(defclass link (rigid-body)
  ((motion-limit-flag
    :initform nil
    :accessor motion-limit-flag)
   (twist-angle
    :initarg :twist-angle
    :accessor twist-angle)
   (link-length
    :initarg :link-length
    :accessor link-length)
   (inboard-joint-angle
    :initarg :inboard-joint-angle
    :accessor inboard-joint-angle)
   (inboard-joint-displacement
    :initarg :inboard-joint-displacement
    :accessor inboard-joint-displacement)
   (inboard-link
    :initarg :inboard-link
    :accessor inboard-link)
   (A-matrix
    :accessor A-matrix)))

(defclass rotary-link (link)
  ((min-joint-angle
    :initarg :min-joint-angle
    :accessor min-joint-angle)
   (max-joint-angle
    :initarg :max-joint-angle
    :accessor max-joint-angle)))

(defclass inverted-pendulum-link (rotary-link)
  ((outboard-link
    :initarg :outboard-link
    :accessor outboard-link)
   (body-coord-angular-acceleration  ;The vector (p-dot q-dot r-dot)
    :initform ’(0 0 0)
    :initarg body-coord-angular-acceleration
    :accessor body-coord-angular-acceleration)
  (posture-translation-rate              ;The vector (xe-dot ye-dot ze-dot).
    :initform (list 0 0 0)
    :initarg :posture-translation-rate
    :accessor posture-translation-rate)
  (posture-translation-acceleration ;The vector (xe-double-dot
                                    ;         ye-double-dot ze-double-dot).
    :initform (list 0 0 0)
    :initarg :posture-translation-acceleration
    :accessor posture-translation-acceleration)
   (body-forces              ;The vector (Fx Fy Fz) in body coordinates.
 :initform (list 0 0 0)
 :initarg :body-forces
 :accessor body-forces)
(body-torques    ;The vector (L M N) in body coordinates.
 :initform (list 0 0 0)
 :initarg :body-torques
 :accessor body-torques)
(equilibrium-torque  ;The torque required to maintain
                     ;     zero rotational acceleration.
 :initform 0
 :initarg :equilibrium-torque
 :accessor equilibrium-torque)
(unit-acceleration-torque  ;The torque required to maintain
                           ;     unity rotational acceleration.
 :initform 0
 :initarg :unit-acceleration-torque
 :accessor unit-acceleration-torque)
(C-column-vector
 :initform 0
 :initarg :C-column-vector
 :accessor C-column-vector)
(equilibrium-inboard-joint-angle
 :initform 0
 :initarg :equilibrium-inboard-joint-angle
 :accessor equilibrium-inboard-joint-angle)
(old-inboard-joint-angle
 :initform 0
 :initarg :old-inboard-joint-angle
 :accessor old-inboard-joint-angle)))
```

```
; **************************************************************************
; File           :human-link.cl
; Author         :Dr. R. McGhee
;                :Naval Postgraduate School
;                :Monterey, CA 93943
; Modified by    :David A. Bretz
; Summary        :Defines foot and individual link classes
;                :Also includes code that rotates rotary links
; **************************************************************************

(defclass foot (rigid-body)
  ((node-list    ;(x y z 1) in body coord for each node. Starts with (0 0 0 1).
    :initform '((0 0 0 1) (0.19047619 0 -0.2285714 1) (-0.495238 0 0.1142857 1)
                (0.3047619 0.0761905 0.1142857 1) (0.3047619 -0.0761905
0.1142857 1)))
   (polygon-list :initform '((1 2 3) (1 3 4) (2 4)))
   ;(motion-limit-flag
   ; :initform nil
   ; :accessor motion-limit-flag)
   (H-matrix :initform (homogeneous-transform '(0 0 0) '(0 0 0)))
   (mass :initform 0.08112)
   (output-vector-to-link1
    :initform '(0 0 0 0 0 0 0 0 0 0 0 0)
    :initarg :output-vector-to-link1
    :accessor output-vector-to-link1)
   (ground-contact-flag
    :initform nil
    :accessor ground-contact-flag)
   (old-euler-angles
    :initform '(0 0 0)
    :initarg :old-euler-angles
    :accessor old-euler-angles)
   ))

(defclass link0 (inverted-pendulum-link)
  ((twist-angle :initform (deg-to-rad -90))
   (link-length :initform 0.19047619)
   (inboard-joint-angle :initform (deg-to-rad 0))
   (inboard-joint-displacement :initform -0.2285714)
   (min-joint-angle :initform (deg-to-rad -360))
   (max-joint-angle :initform (deg-to-rad 360))
   (node-list :initform '((0 0 0 1) (0 0 0 1) (-0.19047619 -0.2285714 0 1)))
   (polygon-list :initform '((1 2)))
   (moment-of-inertia-matrix :initform '((1 0 0) (0 1 0) (0 0 1)))))

(defclass link1 (inverted-pendulum-link)
  ((twist-angle :initform 0)
   (link-length :initform 1.55)
   (inboard-joint-angle :initform (deg-to-rad 90))
   (inboard-joint-displacement :initform 0)
   (min-joint-angle :initform (deg-to-rad 30))
   (max-joint-angle :initform (deg-to-rad 150))
   (node-list :initform '((0 0 0 1) (0 0 0 1) (-1.55 0 0 1)))
   (polygon-list :initform '((1 2)))
   (mass :initform 0.25176)
   (moment-of-inertia-matrix :initform '((1 0 0) (0 0.0372 0) (0 0 1)))
   (output-f-m-to-foot
    :initform '(0 0 0)
    :initarg :output-f-m-to-foot
    :accessor output-f-m-to-foot)
   ))

(defclass link2 (inverted-pendulum-link)
  ((twist-angle :initform 0)
   (link-length :initform 1.447)
   (inboard-joint-angle :initform (deg-to-rad -1))
   (inboard-joint-displacement :initform 0)
   (min-joint-angle :initform (deg-to-rad -90))
   (max-joint-angle :initform (deg-to-rad 2))
   (node-list :initform '((0 0 0 1) (0 0 0 1) (-1.447 0 0 1)))
   (polygon-list :initform '((1 2)))
   (mass :initform 0.5874)
   (moment-of-inertia-matrix :initform '((1 0 0) (0 0.0776 0) (0 0 1)))))

(defclass link3 (inverted-pendulum-link)
  ((twist-angle :initform 0)
   (link-length :initform 2.24)
   (inboard-joint-angle :initform (deg-to-rad 1))
   (inboard-joint-displacement :initform 0)
   (min-joint-angle :initform (deg-to-rad -40))
   (max-joint-angle :initform (deg-to-rad 90))
   (node-list :initform '((0 0 0 1) (0 0 0 1) (-2.24 0 0 1)))
   (polygon-list :initform '((1 2)))
   (mass :initform 3.754)
   (moment-of-inertia-matrix :initform '((1 0 0) (0 1.0 0) (0 0 1)))))

(defmethod update-A-matrix ((link link))
  (with-slots (twist-angle link-length inboard-joint-angle
     inboard-joint-displacement A-matrix) link
    (setf A-matrix (dh-matrix inboard-joint-angle
                              twist-angle
                              link-length
                              inboard-joint-displacement))))

(defmethod rotate ((link link) angle)
  (setf (inboard-joint-angle link) angle)
  (update-A-matrix link)
  (setf (H-matrix link) (matrix-multiply (H-matrix (inboard-link link))
                                         (A-matrix link)))
  (transform-node-list link))

(defmethod rotate-link ((link link) angle)
  (cond ((> angle (max-joint-angle link))
         (rotate link (max-joint-angle link))
         (setf (motion-limit-flag link) t))
        ((< angle (min-joint-angle link))
         (rotate link (min-joint-angle link))
         (setf (motion-limit-flag link) t))
        (t (rotate link angle) (setf (motion-limit-flag link) nil))))
```

```
; ****************************************************************************
; File        :human-model.cl
; Author      :David A. Bretz
;             :Naval Postgraduate School
;             :Monterey, CA 93943
; Summary     :defines the four-link human used for testing in this thesis
; ****************************************************************************

(defclass four-link-human ()
  ((foot
    :initform (make-instance 'foot)
    :accessor foot)
   (link0
    :initform (make-instance 'link0)
    :accessor link0)
   (link1
    :initform (make-instance 'link1)
    :accessor link1)
   (link2
    :initform (make-instance 'link2)
    :accessor link2)
   (link3
    :initform (make-instance 'link3)
    :accessor link3)
   (motion-complete-flag
    :initform nil
    :accessor motion-complete-flag)
   (previous-foot-position
    :initform nil
    :accessor previous-foot-position)
   (current-foot-position
    :initform nil
    :accessor current-foot-position)
   (commanded-horizontal-angle-list
    :initform '(1.7453 1.3963 1.7453)
    :initarg :commanded-horizontal-angle-list
    :accessor commanded-horizontal-angle-list)
   (equilibrium-torque-vector
    :initform '(0 0 0)
    :initarg :equilibrium-torque-vector
    :accessor equilibrium-torque-vector)
   (unit-acceleration-torque-vector
    :initform '(0 0 0)
    :initarg :unit-acceleration-torque-vector
    :accessor unit-acceleration-torque-vector)
   (C-matrix
    :initform '((0 0 0) (0 0 0) (0 0 0))
    :initarg :C-matrix
    :accessor C-matrix)
   (number-of-links
    :initform 3
    :accessor number-of-links)
   (link-list
    :initform '(link1 link2 link3)
    :accessor link-list)
   (horizontal-angle-vector
    :initform '(1.7453 1.3963 1.7453)
    :initarg :horizontal-angle-vector
    :accessor horizontal-angle-vector)
   (angular-velocity-vector
    :initform '(0 0 0)
    :initarg :ANGULAR-VELOCITY-VECTOR
    :accessor ANGULAR-VELOCITY-VECTOR)
   (k-theta
    :initform '(-500 -100 -35)
     :accessor k-theta)
   (k-theta-dot
    :initform '(-500 -100 -30)
    :accessor k-theta-dot)))

(defmethod initialize-human ((human four-link-human))
  (transform-node-list (foot human))
  (setf (inboard-link (link0 human)) (foot human))
  (setf (inboard-link (link1 human)) (link0 human))
  (setf (inboard-link (link2 human)) (link1 human))
  (setf (inboard-link (link3 human)) (link2 human))
  (rotate-link (link0 human) (inboard-joint-angle (link0 human)))
  (rotate-link (link1 human) (inboard-joint-angle (link1 human)))
  (rotate-link (link2 human) (inboard-joint-angle (link2 human)))
  (rotate-link (link3 human) (inboard-joint-angle (link3 human)))
  (setf (outboard-link (link0 human)) (link1 human))
  (setf (outboard-link (link1 human)) (link2 human))
  (setf (outboard-link (link2 human)) (link3 human))
  (setf (outboard-link (link3 human)) (link0 human)) ;set to link0 only for
forces calculation
  (setf (commanded-horizontal-angle-list human)
    (list (inboard-joint-angle (link1 human))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human)))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human))
             (inboard-joint-angle (link3 human)))))
  (setf (horizontal-angle-vector human)
    (list (inboard-joint-angle (link1 human))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human)))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human))
             (inboard-joint-angle (link3 human)))))
  (setf (current-foot-position human)
        (firstn 3 (first (transformed-node-list (link3 human))))))

(defun human-picture ()
  (setf human-1 (make-instance 'four-link-human))
  (initialize-human human-1)
  ;**********
  (setf camera-1 (make-instance 'strobe-camera))
  (camera-window camera-1)
  (setf (enlargement-factor camera-1) 100)
  (take-picture camera-1 human-1))
  ;************
  (setf camera-2 (make-instance 'strobe-camera))
  (camera-window camera-2)
  (setf (title (camera-window camera-2)) "Human - Frontal Plane - Looking
North")
```

```
  (setf (enlargement-factor camera-2) 15)
  (move-body camera-2 '(0 0.2 0) '(-90 0 0))
  (take-picture camera-2 human-1)
  ;*************
  (setf camera-3 (make-instance 'strobe-camera))
  (camera-window camera-2)
  (setf (title (camera-window camera-3)) "Human - Transverse Plane - Looking
Down")
  (setf (enlargement-factor camera-3) 30)
  (move-body camera-3 '(0 -1.57 0) '(0 0 -90))
  (take-picture camera-3 human-1))

 (defun new-picture ()
   (take-picture camera-1 human-1))
   (take-picture camera-2 human-1)
   (take-picture camera-3 human-1))


(defconstant null-move-list '((0 0 0 0 0 0) (0 0 0)))

(defmethod take-picture ((camera strobe-camera) (human four-link-human))
  (take-picture camera (foot human))
  (take-picture camera (link0 human))
  (take-picture camera (link1 human))
  (take-picture camera (link2 human))
  (take-picture camera (link3 human)))

(defmethod move-incremental ((human four-link-human) increment-list)
  (rotate-link (link0 human) 0)
  (rotate-link (link1 human)
               (+ (first increment-list) (inboard-joint-angle (link1 human))))
  (rotate-link (link2 human)
               (+ (second increment-list) (inboard-joint-angle (link2 human))))
  (rotate-link (link3 human)
               (+ (third increment-list) (inboard-joint-angle (link3 human))))
  (setf (previous-foot-position human) (current-foot-position human))
  (setf (current-foot-position human)
        (firstn 3 (first (transformed-node-list (link3 human)))))
  (setf (motion-complete-flag human) (not (or (motion-limit-flag (link1 human))
                                              (motion-limit-flag (link2 human))
                                              (motion-limit-flag (link3
human)))))
  (setf (horizontal-angle-vector human)
    (list (inboard-joint-angle (link1 human))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human)))
          (+ (inboard-joint-angle (link1 human))
             (inboard-joint-angle (link2 human))
             (inboard-joint-angle (link3 human))))))

(defmethod feasible-movep ((human four-link-human) allowable-sinkage allowable-
slippage)
  (and (<= (third (current-foot-position human)) allowable-sinkage)
       (or (minusp (third (current-foot-position human)))
           (minusp (third (previous-foot-position human)))
           (<= (vector-length (vector-slippage human)) allowable-slippage))))
```

```
(defmethod vector-slippage ((human four-link-human))
  (vector-subtract (rest (reverse (previous-foot-position human)))
                   (rest (reverse (current-foot-position human)))))
```

```
; ****************************************************************************
; File      :human-dynamics.cl
; Author    :David A. Bretz
;           :Naval Postgraduate School
;           :Monterey, CA 93943
; Summary   :Contains recursive Newton-Euler inverse and direct dynamics
;           :     algorithm.
; ****************************************************************************

(defmethod update-whole-human ((human four-link-human) delta-time)
  (setf (old-euler-angles (foot human)) (euler-angles (foot human)))
  (update-rigid-body (foot human) nil delta-time)
  (let* ((theta-double-dot (get-angular-acceleration human))
         (theta-dot (angular-velocity-vector human))
         (theta (horizontal-angle-vector human))
         (delta-theta-dot (scalar-multiply delta-time theta-double-dot))
         (delta-theta (vector-add (scalar-multiply delta-time
                                                   (vector-add theta-dot delta-
theta-dot))
                                  (scalar-multiply (* 0.5 (sqr delta-time))
                                                   theta-double-dot)))
         (delta-euler (- (second (old-euler-angles (foot human)))
                         (second (euler-angles (foot human)))))
         (move-list (list (- (first delta-theta) delta-euler)
                          (second delta-theta)
                          (third delta-theta)))
         )
    (setf (angular-velocity-vector human) (vector-add theta-dot delta-theta-
dot))
    ;(setf (horizontal-angle-vector human)
    ;      (vector-add delta-theta (horizontal-angle-vector human)))
    (move-incremental human move-list)
    ;(clear-page (camera-window camera-1))
    (new-picture)
          ))



(defmethod move-away-from-equilibrium ((human four-link-human) offset-list)
  (let* ((move-list offset-list))
    (move-incremental human move-list)
    (new-picture)))

(defmethod stand-up ((human four-link-human) cycles time-step)
  (dotimes (i cycles 'done)
    (update-my-human human-1 time-step)
  ))

(defmethod update-my-human ((human four-link-human) delta-time)
  (let* ((theta-double-dot (get-angular-acceleration human))
         (theta-dot (angular-velocity-vector human))
         (theta (horizontal-angle-vector human))
         (delta-theta-dot (scalar-multiply delta-time theta-double-dot))
         (delta-theta (vector-add (scalar-multiply delta-time (vector-add theta-
dot delta-theta-dot))
                                  (scalar-multiply (* 0.5 (sqr delta-time))
theta-double-dot)))
         (move-list delta-theta)
         (time (time-stamp (link1 human))))
    ;(format t "~A ~%" theta-double-dot)
    (setf (angular-velocity-vector human) (vector-add theta-dot delta-theta-
dot))
    ;(setf (horizontal-angle-vector human) (vector-add delta-theta (horizontal-
angle-vector human)))
    (move-incremental human move-list)
    (setf (time-stamp (link1 human)) (+ time delta-time))
    (plot plotter-2 time (* 100 (- (first (horizontal-angle-vector human))
                                  (first (commanded-horizontal-angle-list
human)))))
    (plot plotter-3 time (* 100 (- (second (horizontal-angle-vector human))
                                   (second (commanded-horizontal-angle-list
human))
                                   )))
    (plot plotter-4 time (* 100 (- (third (horizontal-angle-vector human))
                                   (third (commanded-horizontal-angle-list
human))
                                   )))
    ;(clear-page (camera-window camera-1))
    (new-picture)
          ))

(defmethod get-angular-acceleration ((human four-link-human))
  (post-multiply (matrix-inverse (transpose (build-c-matrix human)))
                 (vector-subtract (get-applied-torque-vector human)
                                  (get-human-torque-vector human '(0 0 0)))))

(defmethod get-applied-torque-vector ((human four-link-human))
  (let* ((theta (horizontal-angle-vector human))
         (theta-goal (commanded-horizontal-angle-list human))
         (theta-dot  (angular-velocity-vector human))
         (k-theta-i (k-theta human))
         (k-theta-dot-i (k-theta-dot human)))
    (vector-add (mapcar '* k-theta-i (vector-subtract theta theta-goal))
                (mapcar '* k-theta-dot-i (vector-subtract theta-dot
                                                          '(0 0 0))))))


(defmethod build-c-matrix ((human four-link-human))
  (let* ((T0 (get-human-torque-vector human '(0 0 0)))
         (T1 (get-human-torque-vector human '(1 0 0)))
         (T2 (get-human-torque-vector human '(0 1 0)))
         (T3 (get-human-torque-vector human '(0 0 1))))
    ;(format t "RUN ~A ~A ~% ~A ~A ~%" T0 T1 T2 T3)
    (list (vector-subtract T1 T0)
          (vector-subtract T2 T0)
          (vector-subtract T3 T0))))

(defmethod get-human-torque-vector ((human four-link-human) angular-
accelerations)
  (let* ((foot-data (output-vector-to-link1 (foot human)))
         (foot-earth-acc (first-three foot-data)) (foot-data (nthcdr 3 foot-
data))
         (foot-angular-acc (first-three foot-data)) (foot-data (nthcdr 3 foot-
data))
```

```
         (foot-angular-velocity (first-three foot-data)) (foot-data (nthcdr 3
foot-data))
         (foot-angle (first-three foot-data))
         (f (get-human-force-vector human angular-accelerations))
         (fx (first f))
         (fz (second f))
         (theta-double-dot-0 (second foot-angular-acc))
         (theta-double-dot-1 (first angular-accelerations))
         (theta-double-dot-2 (second angular-accelerations))
         (theta-double-dot-3 (third angular-accelerations))
         (theta-0 (second foot-angle))
         (theta-1 (first (horizontal-angle-vector human)))
         (theta-2 (second (horizontal-angle-vector human)))
         (theta-3 (third (horizontal-angle-vector human)))
         (fx4 0)
         (fz4 0)
         (lt4 0)
         (data (list lt4 fx4 theta-3 fz4 theta-double-dot-3 (third fx) (third
fz)))
         (lt3 (get-link-torque (link3 human) data))
         (data (list lt3 (third fx) theta-2 (third fz) theta-double-dot-2
(second fx) (second fz)))
         (lt2 (get-link-torque (link2 human) data))
         (data (list lt2 (second fx) theta-1 (second fz) theta-double-dot-1
(first fx) (first fz)))
         (lt1 (get-link-torque (link1 human) data)))
    (setf (output-f-m-to-foot (link1 human)) (list (first fx) (first fz) lt1))
    (list lt1 lt2 lt3)))

(defmethod get-link-torque ((pendulum inverted-pendulum-link) data)
  (let* ((T-i+1 (first data))
         (Fx-i+1 (second data))
         (theta-i (third data))
         (Fz-i+1 (fourth data))
         (length (* 0.5 (link-length pendulum)))
         (J-i (second (second (moment-of-inertia-matrix pendulum))))
         (theta-double-dot-i (fifth data))
         (Fx-i (sixth data))
         (fz-i (seventh data)))
    (+ T-i+1 (* Fx-i+1 length (sin theta-i) -1) (* Fz-i+1 length (cos theta-i))
       (* J-i theta-double-dot-i) (* Fx-i length (sin theta-i) -1) (* Fz-i
length (cos theta-i)))))

(defmethod get-human-force-vector ((human four-link-human) angular-
accelerations)
    (let* ((linear-accelerations (get-human-linear-accelerations human angular-
accelerations))
           (x-double-dot (first linear-accelerations))
           (z-double-dot (second linear-accelerations))
           (mass-0 (mass (foot human)))
           (mass-1 (mass (link1 human)))
           (mass-2 (mass (link2 human)))
           (mass-3 (mass (link3 human)))
           (fx4-fz4 '(0 0))
           (data (list fx4-fz4 mass-3 (third x-double-dot) (third z-double-
dot)))
           (fx3-fz3 (get-link-forces (link3 human) data))
           (data (list fx3-fz3 mass-2 (second x-double-dot) (second z-double-
dot)))
           (fx2-fz2 (get-link-forces (link2 human) data))
           (data (list fx2-fz2 mass-1 (first x-double-dot) (first z-double-
dot)))
           (fx1-fz1 (get-link-forces (link1 human) data))
           )
      (list (list (first fx1-fz1) (first fx2-fz2) (first fx3-fz3))
            (list (second fx1-fz1) (second fx2-fz2) (second fx3-fz3)))
    )
)

(defmethod get-link-forces ((pendulum inverted-pendulum-link) data)
  (let* ((fx-i+1 (first (first data)))
         (fz-i+1 (second (first data)))
         (mass (second data))
         (x-double-dot-i (third data))
         (z-double-dot-i (third data)))
    (list (+ fx-i+1 (* mass x-double-dot-i))
          (+ fz-i+1 (* mass z-double-dot-i) (* mass (third *gravity*) -1)))))

(defmethod get-human-linear-accelerations ((human four-link-human) angular-
accelerations)
  (let* ((foot-data (output-vector-to-link1 (foot human)))
         (foot-earth-acc (first-three foot-data)) (foot-data (nthcdr 3 foot-
data))
         (foot-angular-acc (first-three foot-data)) (foot-data (nthcdr 3 foot-
data))
         (foot-angular-velocity (first-three foot-data)) (foot-data (nthcdr 3
foot-data))
         (foot-angle (first-three foot-data))
         (theta-double-dot-0 (second foot-angular-acc))
         (theta-double-dot-1 (first angular-accelerations))
         (theta-double-dot-2 (second angular-accelerations))
         (theta-double-dot-3 (third angular-accelerations))
         (theta-dot-0 (second foot-angular-velocity))
         (theta-dot-1 (first (angular-velocity-vector human)))
         (theta-dot-2 (second (angular-velocity-vector human)))
         (theta-dot-3 (third (angular-velocity-vector human)))
         (theta-0 (second foot-angle))
         (theta-1 (first (horizontal-angle-vector human)))
         (theta-2 (second (horizontal-angle-vector human)))
         (theta-3 (third (horizontal-angle-vector human)))
         (x0-z0 (list (first foot-earth-acc) (third foot-earth-acc)))
         (data (list x0-z0 theta-0 theta-1 theta-dot-0 theta-dot-1 theta-double-
dot-0 theta-double-dot-1))
         (x1-z1 (link-linear-accelerations (link1 human) data))

         (data (list x1-z1 theta-1 theta-2 theta-dot-1 theta-dot-2 theta-double-
dot-1 theta-double-dot-2))
         (x2-z2 (link-linear-accelerations (link2 human) data))

         (data (list x2-z2 theta-2 theta-3 theta-dot-2 theta-dot-3 theta-double-
dot-2 theta-double-dot-3))
         (x3-z3 (link-linear-accelerations (link3 human) data))
         )
    (list (list (first x1-z1) (first x2-z2) (first x3-z3))
          (list (second x1-z1) (second x2-z2) (second x3-z3)))
    ))

(defmethod link-linear-accelerations ((pendulum inverted-pendulum-link) data)
  (let* ((x-double-dot-i-1 (first (first data)))
         (z-double-dot-i-1 (second (first data)))
         (theta-i-1 (second data))
         (theta-i (third data))
         (theta-dot-i-1 (fourth data))
         (theta-dot-i (fifth data))
         (theta-double-dot-i-1 (sixth data))
         (theta-double-dot-i (seventh data)))
    (list (- x-double-dot-i-1
             (* (* (link-length (inboard-link pendulum)) 0.5)
                (+ (* theta-double-dot-i-1 (sin theta-i-1))
                   (* (sqr theta-dot-i-1) (cos theta-i-1))))
             (* (* (link-length pendulum) 0.5)
                (+ (* theta-double-dot-i (sin theta-i))
                   (* (sqr theta-dot-i) (cos theta-i)))))
          (- (* z-double-dot-i-1 -1)
             (* (* (link-length (inboard-link pendulum)) 0.5)
                (+ (* theta-double-dot-i-1 (cos theta-i-1))
                   (* (sqr theta-dot-i-1) (sin theta-i-1))))
             (* (* (link-length pendulum) 0.5)
                (+ (* theta-double-dot-i (cos theta-i))
                   (* (sqr theta-dot-i) (sin theta-i))))))))

(defun sqr (x)
  (* x x))
```

```
; ****************************************************************************
; File      :foot-dynamics.cl
; Author    :David A. Bretz
;           :Naval Postgraduate School
;           :Monterey, CA 93943
; Summary   :Calculates the forces and moments acting on the foot.
;           :Called from the euler-angle-rigid-body code.
;           :Also defines the ground class.
; ****************************************************************************

(defclass ground (rigid-body)
  ((node-list
    :initform '((0 0 0 1) (0 0 0 1) (5 0 0 1) (-5 0 0 1) (0 5 0 1) (0 -5 0 1)))
   (polygon-list :initform '((1 2) (1 3) (1 4) (1 5) (2 4 3 5)))
   ))

(defun forces-moments (state time)
  (let* ((z (third state))
         (theta (fifth state))


         (location (first-three state)) (state (nthcdr 3 state))
         (angles (first-three state)) (state (nthcdr 3 state))
         (uvw (first-three state)) (state (nthcdr 3 state)) (pqr state)
         (R (rotation-matrix angles)) (v-earth (post-multiply R uvw))
         (M (body-rate-to-euler-rate-matrix angles))
         (euler-angle-rates (post-multiply M pqr))

         (fx1 (first (output-f-m-to-foot (link1 human-1))))
         (fz1 (second (output-f-m-to-foot (link1 human-1))))
         (t1 (third (output-f-m-to-foot (link1 human-1))))

         (u (first uvw))
         (w (third uvw))
         (xe-dot (first v-earth))
         (ze-dot (third v-earth))
         (theta-dot (second euler-angle-rates))

         (torsional-damening-const 0)
         (spring-const 10000)
         (dampening-const 100)
         (ze-toe (+ z (* 0.508253869 (sin (+ theta 0.226798848054)))))
         (ze-heel (+ z (* 0.325485857 (sin (- 0.358770670271 theta)))))
         (fze-toe (cond ((>= ze-toe 0) (+ (* spring-const ze-toe -1)
                                          (* dampening-const ze-dot -1)))
                        ((< ze-toe 0) 0)))
         (fze-heel (cond ((>= ze-heel 0) (+ (* spring-const ze-heel -1)
                                            (* dampening-const ze-dot -1)))
                         ((< ze-heel 0) 0)))
         (fx-body (+ (* fze-toe (sin theta) -1)
                     (* fze-heel (sin theta))
                     ;(* dampening-const xe-dot 0)
                     (* fx1 (sin theta) -1)))
         (fz-body (+ (* fze-toe (cos theta))
                     (* fze-heel (cos theta))
                     (* fz1 (cos theta) -1)))
         (m-body (+ (* fx-body 0.1142857); includes fx1 but at wrong distance!
                    (* fze-toe (cos theta) 0.495238)
                    (* fze-heel (sin theta) 0.3047619); cos
                    (* torsional-damening-const theta-dot -1)
                    (* t1 -1)))
         (fz-ground (+ (* fze-toe (cos theta))
                       (* fze-heel (cos theta))))


         )
    ;(format t "state ~A ~%" (append location angles uvw pqr))
    (format t "fz-ground ~A    ~%" fz-ground)
    (format t "forces ~A ~A ~A ~A ~A ~%" fze-toe fze-heel fx-body fz-body m-
body)
    (setf (ground-contact-flag (foot human-1)) (cond ((or (>= ze-toe 0) (>= ze-
heel 0)) t)))
    (plot plotter-1 time (- fz-ground))
    (list fx-body 0 fz-body 0 m-body 0)))
```

```
; *****************************************************************************
; File      :test.cl
; Author    :David A. Bretz
;           :Naval Postgraduate School
;           :Monterey, CA 93943
; Summary   :top-level functions for
;           :1)testing the foot alone
;           :2)testing the postural control
;           :3)testing the four-link-human
; *****************************************************************************

(defun test-foot ()
  (setf foot-1 (make-instance 'foot))
  (setf ground-1 (make-instance 'ground))
  (setf camera-1 (make-instance 'strobe-camera))
  (setf plotter-1 (make-instance 'plotter))
  (draw-coordinate-axes (plotter-window plotter-1))
  (initialize foot-1)
  (initialize ground-1)
  (move foot-1 '(0 0 0) '(0 0 -0.121))
  (setf (enlargement-factor camera-1) 100)
  (take-picture camera-1 foot-1)
  (take-picture camera-1 ground-1)
  (dotimes (i 100 'done) (update-rigid-body foot-1 nil .001))
  (take-picture camera-1 foot-1)
  (dotimes (i 1 'done)
    (time-stamp foot-1)
    (dotimes (i 100 'done) (update-rigid-body foot-1 nil .001))
    (take-picture camera-1 foot-1)
    (time-stamp foot-1)
    )
  )

(defun test-posture ()
  (human-picture)
  (setf plotter-2 (make-instance 'plotter))
  (draw-coordinate-axes (plotter-window plotter-2))
  (setf plotter-3 (make-instance 'plotter))
  (draw-coordinate-axes (plotter-window plotter-3))
  (setf plotter-4 (make-instance 'plotter))
  (draw-coordinate-axes (plotter-window plotter-4))
  (move-away-from-equilibrium human-1 '(0 0 0)) ;delta change of vertical theta
  (stand-up human-1 1000 0.001) ;cycles and time-step
  )

(defun big-test ()
  (human-picture)
  (setf ground-1 (make-instance 'ground))
  (setf plotter-1 (make-instance 'plotter))
  (draw-coordinate-axes (plotter-window plotter-1))
  (initialize ground-1)
  (take-picture camera-1 ground-1)
  (move (foot human-1) '(0 0 0) '(0 0 -0.1142857))
  (move-incremental human-1 '(0 0 0))
  ;(move (foot human-1) '(0 -0.32175 0) '(0 0 -0.2376))
  ;(move-incremental human-1 '(-0.02702 0 0.4))
  ;(setf (angular-velocity-vector human-1) '(-.973 -.973 -.973))
  ;(setf (body-coord-angular-velocity (link1 human-1)) (list 0 -.973 0))
```

```
;(setf (body-coord-linear-velocity (link1 human-1)) (list 0 0 0))
;(setf (body-coord-angular-velocity (link2 human-1)) (list 0 -.5 0))
;(setf (body-coord-linear-velocity (link2 human-1)) (list -2.0 0 0))
;(setf (body-coord-angular-velocity (link3 human-1)) (list 0 0 0))
;(setf (body-coord-linear-velocity (link3 human-1)) (list -4.5 0 0))
(dotimes (i 400 'done)
  (update-whole-human human-1 .0001)
  )
)
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Abdelnour, T. A., Passerello, C. E., and Huston, R. L., “An Analytical Analysis of Walking,” ASME (Paper) no. 75-WA/Bio-4, Nov. 30-Dec. 4, 1975.

Abu-Faraj, Z. O., Harris, G. F., Chang, A., Shereff, M. J., and Nery, J., “Quantitative Evaluation of Plantar Pressure Alterations with Metatarsal and Scaphoid Pads,” Chapter 20 of *Human Motion Analysis: Current Applications and Future Directions,* Harris, G. F. and Smith, P. A., eds., IEEE Press, New York, 1996.

Amirouche, F. M. L., Ider, S. K., and Trimble, J., "General analytical method for the analysis and simulation of human locomotion," *American Society of Mechanical Engineers, Bioengineering Division,* (Publication) BED, vol. 8, pp. 57-60, Nov. 27 - Dec. 2, 1988.

Bates, B. T., DeVita, P., and Lander, J. E., "Reliability of Ground Reaction Force Data," Proceedings of the 37th Annual Conference on Engineering in Medicine and Biology, Los Angeles, CA, vol. 26, pp. 95, 1984.

Bediz, M., *A Computer Simulation Study of a Single Rigid Body Dynamic Model for Biped Postural Control*, Masters Thesis, Naval Postgraduate School, Monterey, CA, March, 1997.

Bresler, B. and Frankel, J. P., “The Forces and Moments in the Leg During Level Walking,” *Transactions of the ASME*, pp. 27-36, Jan., 1950.

Cavanagh, P. R., Williams, K. R., and Clarke, T. E., “Comparison Of Ground Reaction Forces During Walking Barefoot And In Shoes,” *Biomechanics 7B*, Proceedings of the 7th International Congress on Biomechanics. Warsaw, Poland, Univ. Park Press Baltimore, MD, USA, pp. 151-156, 1981.

Chao, E. Y., Laughman, R. K., Schneider, E., and Stauffer, R.N., “Normative Data of Knee Joint Motion and Ground Reaction Forces in Adult Level Walking,” *Journal of Biomechanics*, vol. 16, no. 3, pp. 219-233, 1983.

Collins, J. J., and Whittle, M. W., “Impulsive Forces During Walking And Their Clinical Implications,” *Clinical Biomechanic,* vol. 4, no. 3, pp. 179-187, Aug., 1989a.

Collins, J. J. and Whittle, M. W., “Influence of Gait Parameters on the Loading of the Lower Limb,” *Journal of Biomedical Engine*ering, vol. 11, no. 5, pp. 409-412, Sep., 1989b.

Craig, J., *Introduction to Robotics: Mechanics and Control,* Second Edition, Addison-Wesley Publishing Company, Inc., Menlo Park, CA, 1989.

D'Andrea, S. E., Lord, D. R., and Davis, B. L., "A Rheological Model of the Human Heel Pad," Presented at the 21st Annual Meeting of the American Society of Biomechanics, Clemson, Univ., SC, USA, Sep. 24-27, 1997.

Davidson, S., *An Experimental Comparison of CLOS and C++ Implementations of an Object-Oriented Graphical Simulation of Walking Robot Kinematics*, Masters Thesis, Naval Postgraduate School, Monterey, CA, March, 1993.

Davis, R. B. and DeLuca, P. A., "Clinical Gait Analysis: Current Methods and Future Directions," Chapter 2 of *Human Motion Analysis: Current Applications and Future Directions,* Harris, G. F. and Smith, P. A., eds., IEEE Press, New York, 1996.

DeMaio, Marlene, CDR, USN, "Development of an Improved Deck and Ladder Surface for U.S. Naval Ships: A Method to Decrease Knee Injuries," Proposal to the Secretary of the Navy: Improved Deck and Ladder Surface, unpublished, 1999.

Dingwell, J. B., Lloyd, T., and Cavanagh, P. R., "Quantifying Ten-hour Load Bearing Activities in a Group of Adolescent Women," Presented at the 21st Annual Meeting of the American Society of Biomechanics, Clemson, Univ., SC, USA, Sep. 24-27, 1997.

Graham, p., *ANSI Common Lisp*, Prentice Hall, New Jersey, 1996.

Guilak, F., Sah, R., and Setton, L. A., "Physical Regulation of Cartilage Metabolism," Published in *Basic Orthopaedic Biomechanics, 2nd ed.,* Mow, V. C. and Hayes, W. C., eds., Lippincott-Raven, Philadelphia, 1997.

Hansen, L., Winkel, J., and Jørgensen, K., "Significance of mat and shoe softness during prolonged work in upright position: based on measurements of low back muscle EMG, foot volume changes, discomfort and ground force reaction," *Applied Ergonomics*, vol. 29, no. 3 pp. 217-224, 1998.

Helmkamp, J. C. and Coben,P. A., "Knee Injuries and Disability Among Enlisted Males in the U.S. Navy," 16 Pages. AD Number: ADA160936 Corporate Author: Naval Health Research Center, San Diego, CA, May 1985.

Hoppenfeld, S. and Zeide, M. S., *Orthopaedic Dictionary*, Lippincott-Raven, New York, 1994.

Ju, M. and Mansour, J. M., "Simulation of the Double Limb Support Phase of Human Gait," *Journal of Biomechanical Engineering*, vol. 110, no. 8, pp. 223-229, 1988.

Kerrigan, D. C., Todd, M. K., and Croce, U. D., "Gender Differences in Joint Biomechanics During Walking: Normative Study in Young Adults," *American Journal of Physical Medicine & Rehabilitation,* vol. 77, no. 1, Williams & Wilkins Co. Baltimore, MD, USA, pp. 2-7, Jan-Feb., 1998.

King, A. I. And Nakhla S., "Lower Limb Biomechanics," *Biomechanics and Medical Aspects of Lower Limb Injuries*, SAE, Warrendale, PA, Oct., 1986.

Knudson, L. E., *A Three-dimensional Computer Model of Whole-body Human Motion*, Ed.D. Dissertation, University of Northern Colorado, 1980.

Koozekanani, S. H., Barin, K., McGhee, R. B., and Chang, H. T., "A Recursive Free-Body Approach to Computer Simulation of Human Postural Dynamics," *IEEE Transactions on Biomedical Engineering*, vol. BME-30, no. 12, Dec., 1983.

Macellari, V., Giacomozzi, C., Scattolini, E., and Cappozzo, A., "Piezo-Dynamometry Of Foot-To-Floor Interactions During Locomotion," *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings,* Proceedings of the 1996 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Part 2 (of 5) Amsterdam, Neth., IEEE Piscataway, NJ, USA, vol. 2, pp. 465-466, 1996.

McGhee, R. B., Koozekanani, S. H., Weimer, F. C., and Rahmani, S., "Dynamic Modeling of Human Motion," *Proceedings of Joint Automatic Control Conference*, Denver, CO, pp. 405-413, 1979.

McGhee, R. B., "Computer Simulation of Human Movements," *Biomechanics of Motion*, Morecki, A., ed., CISM Courses and Lectures, No. 263, Springer-Verlag Wien, New York, 1980.

McMahon, T. A. and Greene, P. R., "The influence of track compliance on running," *Journal of Biomechanics,* vol. 12, pp. 893-904, 1979.

Morecki, A., Olszewski, J., Jaworek, K, McGhee, R. B., Koozekanani, S. H., and Burnett, C. N., "Automatic Computer Analysis of Human Gait," *Biomechanics 7B*, Proceedings of the 7th International Congress on Biomechanics, Warsaw, Poland, Univ. Park Press, Baltimore, MD, USA, pp. 133-140, 1981.

Pandy, M. G., *Models for Understanding the Dynamics of Human Walking*, PhD Dissertation, Ohio State University, 1987.

Pandy, M. G., Berme, N., "Numerical Method For Simulating The Dynamics Of Human Walking," *Journal of Biomechanics,* vol. 21, no. 12, pp. 1043-1051,1988.

Perry, J., *Gait Analysis, Norman and Pathological Function*, SLACK Inc., New Jersey, 1992.

Radin, E. L., Orr, R. B., Keman, J. L., Paul, I. L. and Rose, R. M., "Effect of Prolonged Walking on Concrete on the Knees of Sheep," *Journal of Biomechanics*, vol. 15, pp. 487-92, 1982.

Röhrle, H., Scholten, R., Sigolotto, C., Sollbach, W., and Kellner, H., "Joint Forces in the Human Pelvis-Leg Skeleton During Walking," *Journal of Biomechanics*, vol. 17, no. 6, pp. 409-424, 1984.

Ross, R. F. and Switzer, W. P., "Mycoplasmal Arthritis of Swine," *Medical Clinics of North America*, vol. 52 pp. 677-86, 1968.

Siegler, S., Seliktar, R., and Hyman, W., "Simulation Of Human Gait With The Aid Of A Simple Mechanical Model," *Journal of Biomechanics,* vol. 15, no. 6, pp. 415-425, 1982.

Simon, S. R., Paul, I. L. Mansour, J., Munro, M., Abernethy, P. J. and Radin, E. L., "Peak Dynamic Force in Human Gait," *Journal of Biomechanics*, vol. 14, no. 12, pp. 817-822, Dec. 1981.

Stepanenko, Y. and Vukobratovic, M., "Dynamics of Articulated Open-chain Active Mechanisms," *Mathematical Biosciences*, vol. 29, pp. 137-170, 1976.

Verstraete, M. C., *Method for Computing the Three-Dimensional Forces and Moments in the Lower Limb During Locomotion*, Ph.D. Dissertation, Michigan State University, 1988.

Voloshin, A. S. and Wosk, J., "Influence of Artificial Shock absorbers on Human Gait," *Clinical Orthopedics and Related Research*, vol. 160, pp. 52-6, 1981.

Wilson, D. R., MacWilliams, B. A., DesJardins, J. D., and Chao, E. Y. S., "In Vitro Simulation of Knee Joint Mechanics for the Validation of Biomechanical Models," *American Society of Mechanical Engineers, Bioengineering Division,* (Publication) BED, Advances in Bioengineering Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exposition, Nov. 17-22, 1996, Atlanta, GA, USA, vol. 33, pp. 323-324, 1996.

Wosk, J. and Voloshin, A. S., "Low Back Pain: Conservative Treatment with Artificial Shock Absorbers," *Archives of Physical Medicine and Rehabilitation*, vol. 66 pp. 145-8, 1985.

Zanchi, V. and Cecic, M., "Simulation of human locomotion system," *International Journal for Engineering Modelling,* vol. 9, no. 1-4, pp. 51-54, 1996.

BIBLIOGRAPHY

Brodland, G. W. and Thornton-Trump, A. B., "Gait Reaction Reconstruction and a Heel Strike Algorithm," *Journal of Biomechanics*, vol. 20, no. 8, pp. 767-772, 1987.

Charteris, J., "Comparison of the effects of backpack loading and of walking speed on foot-floor contact patterns," *Ergonomics*, Taylor & Francis Ltd., London, England, vol. 41, no. 12, pp. 1792-1809, Dec. 1998.

Chaffin, D. B., *Occupational Biomechanics*, J. Wiley and Sons, Inc., New York, 1991.

Chen, L., Raftopoulos, D. D., and Armstrong, C. W., "Analysis of Gait Mechanics by Vector Diagrams," *American Society of Mechanical Engineers, Bioengineering Division*, (Publication) BED 1991, Advances in Bioengineering Winter Annual Meeting of the American Society of Mechanical Engineers, Dec 1-6, 1991, vol. 20, pp. 187-189, 1991.

Denavit J. and Hartenberg, R. S. "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Journal of Applied Mechanics*, vol. 22, no. 2, pp. 215-221, June 1955.

Duda, G. N., Schneider, E., and Chao, E. Y. S. "Internal Forces and Moments in the Femur During Walking," *Journal of Biomechanics*, vol. 30, no. 9, pp. 933-941, Sep. 1997.

Engin, A. E., and Moeinzadeh, M. H., *Modeling of Human Joint Structures*, ADA121076, 180 Pages, Sep. 1982.

Fung, Y. C., *Biomechanics: Mechanical Properties of Living Tissues*, Springer-Verlag, New York, 1993.

Gil, J., Li, G., Kanamori, A., and Woo, S. L.-Y., "Development of a 3D Computational Human Knee Joint Model," *American Society of Mechanical Engineers, Bioengineering Division*, (Publication) BED, Advances in Bioengineering Proceedings of the 1998 ASME International Mechanical Engineering Congress and Exposition, Nov 15-20 1998, Anaheim, CA, USA, vol. 39, 1998.

Grandjean, E., *Fitting the Task to the Man; An Ergonomic Approach*, Taylor and Francis Ltd., London, 1975.

Gray, H., *Gray's Anatomy*, Bounty Books, New York, 1977.

Gray, J., *Animal Locomotion*, W. W. Norton and Co. Inc., New York, 1968.

Hall, S. J., *Basic Biomechanics*, WCB/McGraw-Hill, New York, 1999.

Hardt, D. E., "Determining Muscle Forces in the Leg During Normal Human Walking - An Application and Evaluation of Optimization Methods," *Journal of Biomechanical Engineering*, vol. 100, pp. 72-78, May 1978.

Harrington, I. J., "Bioengineering Analysis of Force Actions at the Knee in Normal and Pathological Gait," *Bio-Medical Engineering* (London), vol. 11, no. 5, pp. 167-172, May, 1976.

Harris, G. F. and Smith, P. A., *Human Motion Analysis: Current Applications and Future Directions*, IEEE, New York, 1996.

Hemami, H. and Wyman, B. F. "Indirect Control of the Forces of Constraint in Dynamics Systems," *Journal of Dynamic Systems, Measurements, and Control*, vol. 101, pp. 355-360, Dec. 1979.

Hurwitz, D. E., Sumner, D. R., Andriacchi, T. P., and Sugar, D. A., "Dynamic Knee Loads During Gait Predict Proximal Tibial Bone Distribution," *Journal of Biomechanics*, vol. 31, no. 5, pp. 423-430, May, 1998.

Johnson, F., Scarrow, P., and Waugh, W., "Assessments of Loads in the Knee Joint," *Medical & Biological Engineering & Computing*, vol. 19, no. 2, pp. 237-243, Mar. 1981.

Keller, T. S., Weisberger, A. M., Ray, J. L., Hasan, S. S., Shiavi, R. G., and Spengler, D. M. "Relationship Between Vertical Ground Reaction Force and Speed During Walking, Slow Jogging, And Running," *Clinical Biomechanics*, vol. 11, no. 5, pp. 253-259, Jul, 1996.

King, A. I., "A Review of Biomechanical Models," *Journal of Biomechanical Engineering*, vol. 106, pp. 97-104, May 1984.

Kljajic, M. and Krajnik, J. "Use of Ground Reaction Measuring Shoes in Gait Evaluation," *Clinical Physics and Physiological Measurement*, vol. 8, no. 2, pp. 133-142, May, 1987.

Komistek, R. D., Stiehl, J. B., Dennis, D. A., Paxson, R. D., and Soutas-Little, R. W., "Mathematical Model of the Lower Extremity Joint Reaction Forces Using Kane's Method of Dynamics," *Journal of Biomechanics*, vol. 31, no. 2, pp. 185-189, Feb. 1998.

Koozekanani, S. H., Stockwell, C. W., McGhee, R. B., and Firoozmand, F., "On the Role of Dynamic Models in Quantitative Posturography," *IEEE Transactions on Biomedical Engineering*, vol. BME-27, no. 10, Oct. 1980.

Lucas, G. L., Cooke, F. W., and Friis, E. A., *A Primer of Biomechanics*, Springer-Verlag, New York, 1999.

Luh, J. Y. S., Walker, M. W., and Paul, R. P. C., "On-Line Computational Scheme for Mechanical Manipulators," *Journal of Dynamic Systems, Measurements, and Control*, vol. 102, pp. 69-76, June 1980.

Mikosz, R. P., Andriacchi, T. P., and Andersson, G. B. J., "Model Analysis of Factors Influencing the Prediction of Muscle Forces at the Knee," *Journal of Orthopaedic Research*, vol. 6, no. 2, pp. 205-214, Mar. 1988.

Morecki, A., Ekiel, J., and Fidelus, K., *Cybernetic Systems of Limb Movements in Man, Animals and Robots*, Ellis Horwood Limited, New York, 1984.

Nisell, R., "On the Biomechanics of the Knee - A Study of Joint and Muscle Load with Applications in Ergonomics, Orthopaedics and Rehabilitation *Clinical Biomechanics*, vol. 1, no. 2, pp. 90, May, 1986.

Nordin, M. and Frankel, V. H., *Basic Biomechanics of the Musculoskeletal System*, New York, 1989.

Orin, D. E. and Oh, S. Y., "Control of Force Distribution in Robotic Mechanisms Containing Closed Kinematic Chains," *Journal of Dynamic Systems, Measurements, and Control*, vol. 102, pp. 134-141, June 1981.

Pandy, M. G. and Berme, N., "Synthesis of Human Walking: A Planar Model for Single Support," *Journal of Biomechanics*, vol. 21, no. 12, pp. 1053-1060, 1988.

Pandy, M. G. and Berme, N., "Quantitative Assessment of Gait Determinants During Single Stance via a Three-dimensional Model. Part 1. Normal Gait," *Journal of Biomechanics*, vol. 22, no. 6-7, pp. 717-724, 1989.

Radin, E.L., Orr, R.B., Schein, S.L., Keman, J.L., and Rose, R.M. "The Effects of Hard and Soft Surface Walking on Sheep Knees," *Journal of Biomechanics* vol. 13., pp. 196, 1980.

Scheiner, A., Ferencz, D. C., and Chizeck, H. J. "Simulation of Normal and FES Induced Gait Using a 23 Degree-of-freedom Model," *Proceedings of the Annual Conference on Engineering in Medicine and Biology*, IEEE, vol. 15, pt. 3, pp. 1155-1156, Oct. 28-31, 1993.

Shelburne, K. B. and Pandy, M. G., "Musculoskeletal Model of the Knee for Evaluating Ligament Forces During Isometric Contractions," *Journal of Biomechanics*, vol. 30, no. 2, pp. 163-176, Feb. 1997.

Smothers, C., Ray, J., and Wildman, G., "Comparison of Heel Strike Accelerations While Walking on Carpet, Tile, and a Motorized Treadmill," *Critical Reviews in Biomedical Engineering*, vol. 26, no. 5, pp. 342, 1998.

Thomas, Clayton L., *Taber's Cyclopedic Medical Dictionary*, F. A. Davis Company, Philadelphia, 1981.

Thornton-Trump, A. B. and Daher, R., "Prediction of Reaction Forces from Gait Data," *Journal of Biomechanics*, vol. 8, no. 3-4, pp. 173-178, Jul. 1975.

Tortora, G. J. and Anagnostakos, N. P., *Principles of Anatomy and Physiology*, Harper & Row, New York, 1984.

Walker, M. W. and Orin, D. E., "Efficient Dynamic Computer Simulation of Robotic Mechanisms," *Journal of Dynamic Systems, Measurements, and Control*, vol. 104, pp. 205-211, Sep. 1982.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
Naval Postgraduate School
411 Dyer Rd.
Monterey, California 93943-5101

3. Engineering & Technology Curricular Office (Code 34) 1
Naval Postgraduate School
Monterey, California 93943

4 Computer Science Curricular Office (Code 32).. 1
Naval Postgraduate School
Monterey, California 93943

5. Professor Young W. Kwon, Code ME/KW 2
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California 93943

6. Professor Robert B. McGhee, Code CS/MZ 1
Department of Computer Science
Naval Postgraduate School
Monterey, California 93943

7. CDR Marlene DeMaio, MD, USN 1
Department of Orthopaedic Surgery
National Naval Medical Center
Bethesda, Maryland 20889

8. David A. Bretz, LT, USN.. 2
2120 E. County Rd. 1350 N.
Newman, Illinois 61942